INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

Weekly Progress Report

Project: TinyML-Based Project on FPGA Board with RISC-V Core

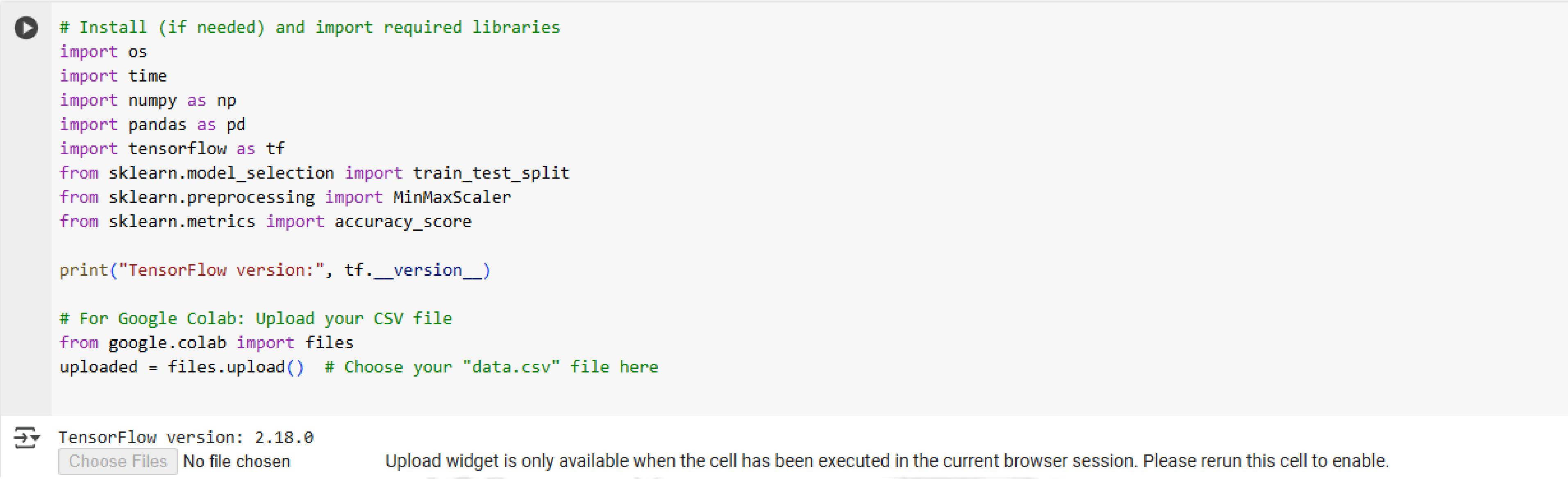
Name: Aman Chauhan
Branch: Btech. Computer Science and Engineering (CSE Core)
Roll Number: 22BCE0476
College: Vellore Institute of Technology, Vellore
Mentor: Dr. Sudip Roy

Week: *5th report*
Date: *20th June 2025*

Summary of Week 5 Activities

- Developed and trained AI models using interpreted sensor, audio, and image data, then converted them into optimized **TensorFlow Lite (.tflite)** formats
- Utilized Edge Impulse for building and deploying **audio classification**, **object recognition**, and **object detection** models with real-time processing capabilities.
- Tested models on various embedded boards by following **Edge Impulse documentation**, ensuring efficient on-device performance across multiple platforms.

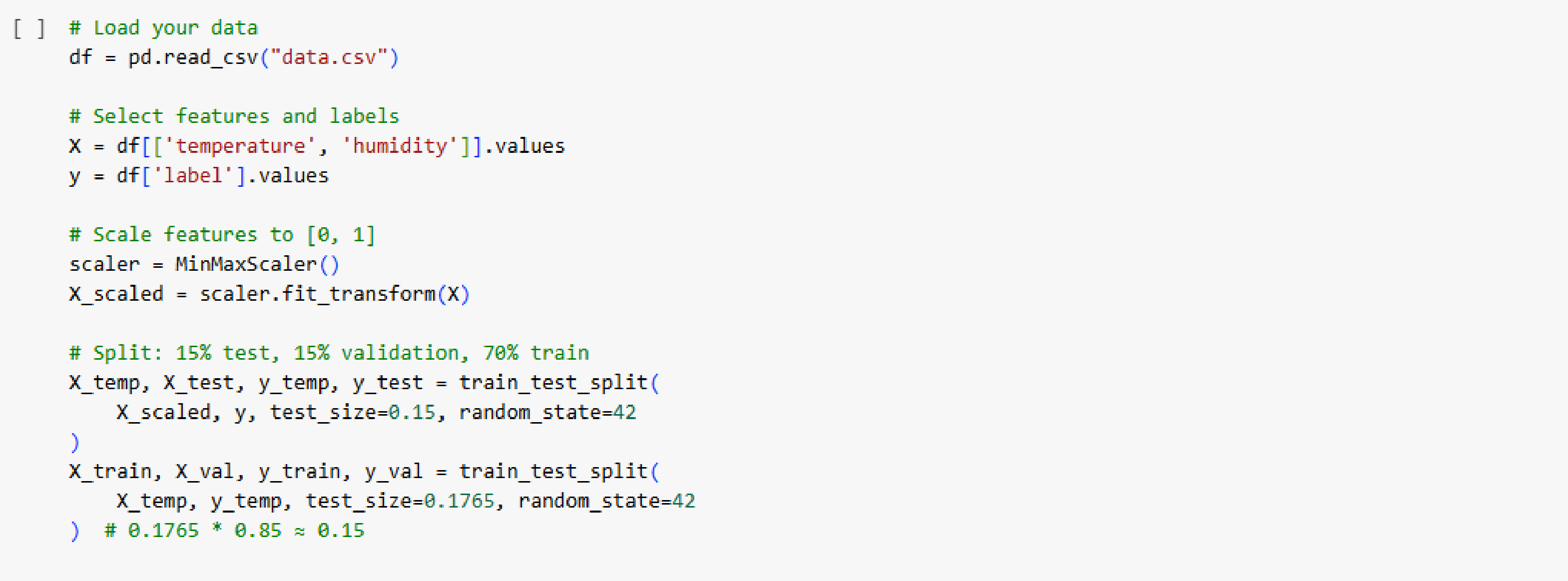
Google Collab Code (TFLite Model)



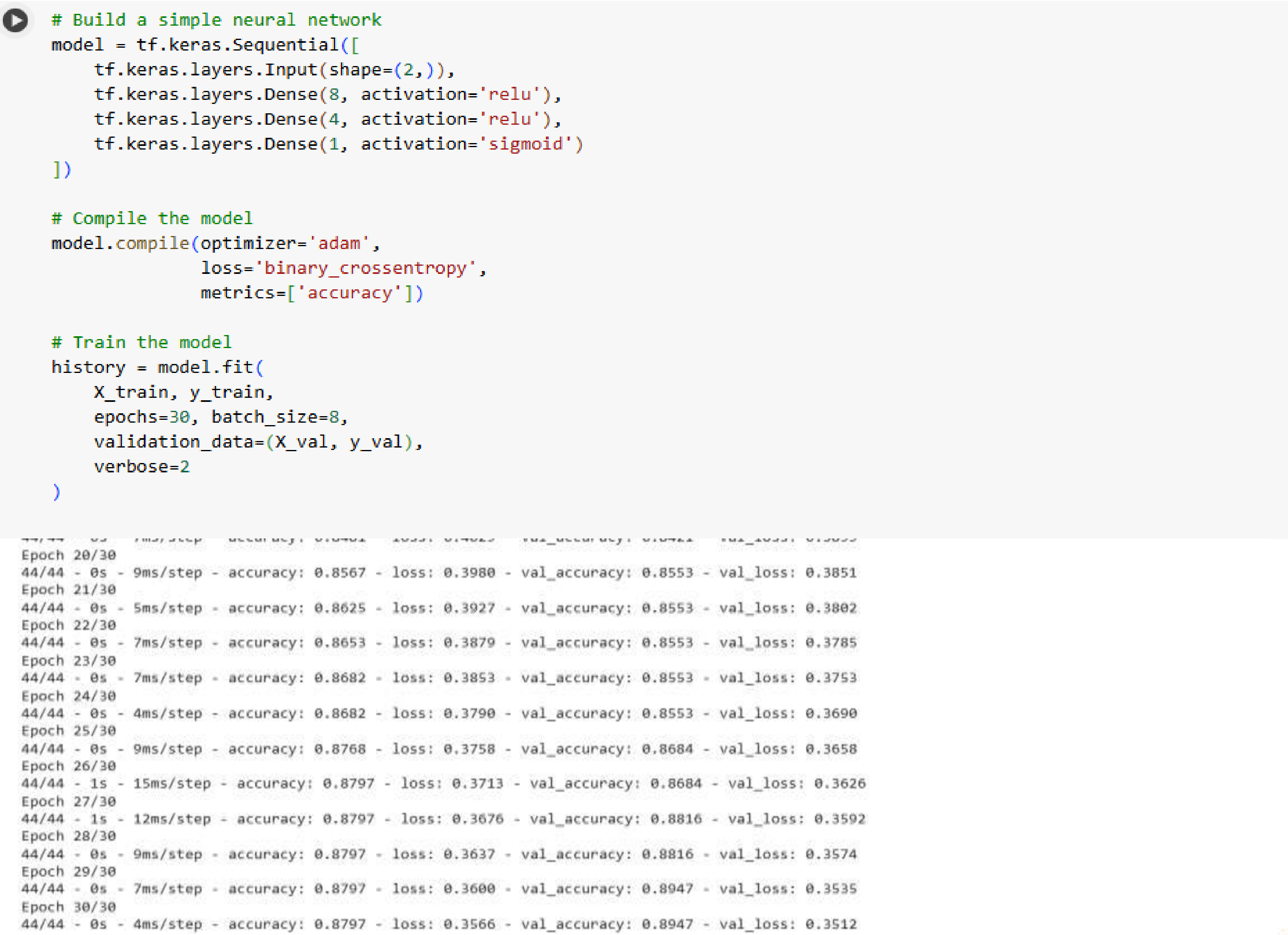
```python
# Install (if needed) and import required libraries
import os
import time
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score

print("TensorFlow version:", tf.__version__)

# For Google Colab: Upload your CSV file
from google.colab import files
uploaded = files.upload()  # Choose your "data.csv" file here
```

```
TensorFlow version: 2.18.0
```

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

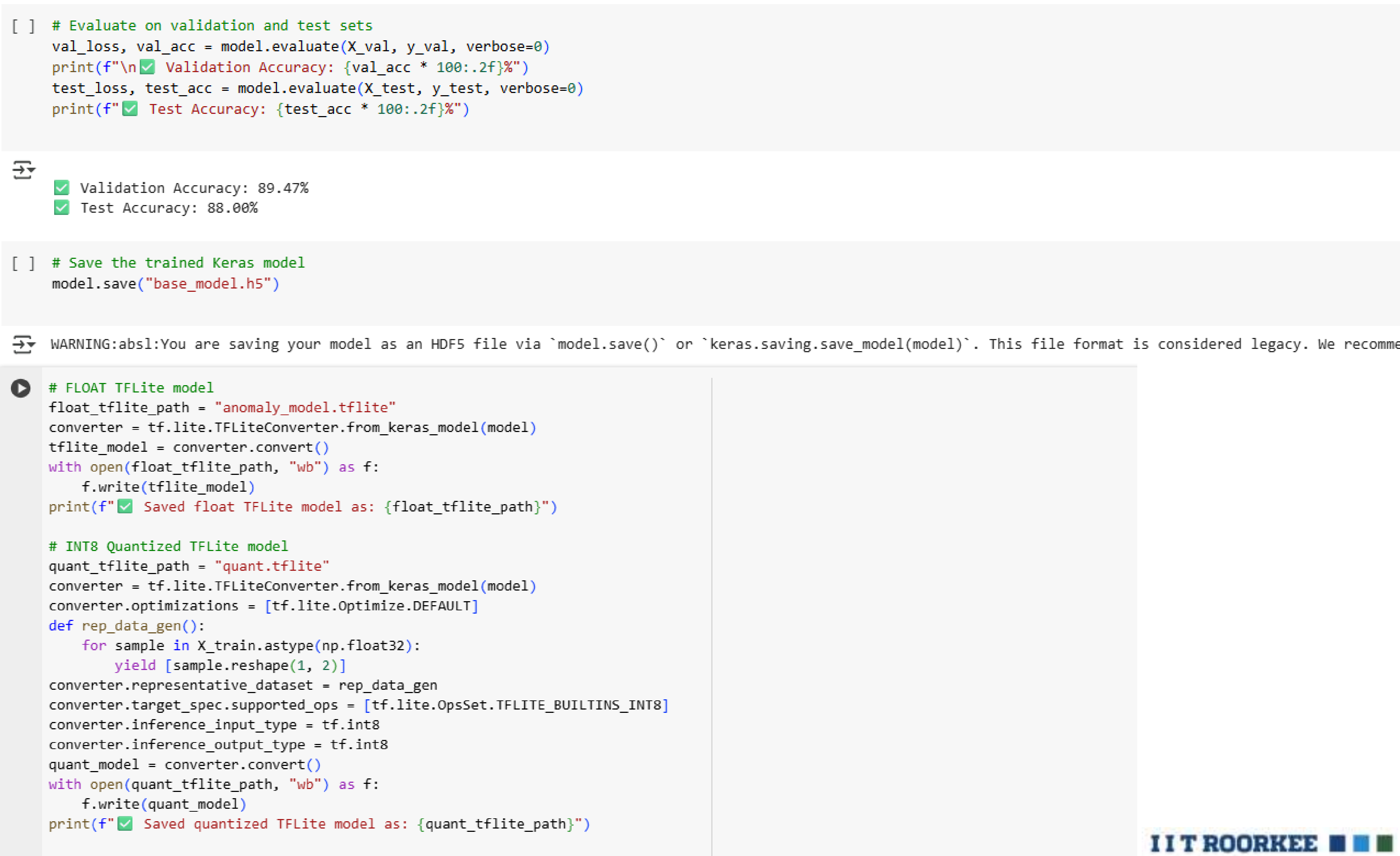


```python
# Load your data
df = pd.read_csv("data.csv")

# Select features and labels
X = df[['temperature', 'humidity']].values
y = df['label'].values

# Scale features to [0, 1]
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Split: 15% test, 15% validation, 70% train
X_temp, X_test, y_temp, y_test = train_test_split(
    X_scaled, y, test_size=0.15, random_state=42
)
X_train, X_val, y_train, y_val = train_test_split(
    X_temp, y_temp, test_size=0.1765, random_state=42
)  # 0.1765 * 0.85 ≈ 0.15
```



```
# Build a simple neural network
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(2,)),
    tf.keras.layers.Dense(8, activation='relu'),
    tf.keras.layers.Dense(4, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(
    X_train, y_train,
    epochs=30, batch_size=8,
    validation_data=(X_val, y_val),
    verbose=2
)
```

```
Epoch 20/30
44/44 - 0s - 9ms/step - accuracy: 0.8567 - loss: 0.3980 - val_accuracy: 0.8553 - val_loss: 0.3851
Epoch 21/30
44/44 - 0s - 5ms/step - accuracy: 0.8625 - loss: 0.3927 - val_accuracy: 0.8553 - val_loss: 0.3802
Epoch 22/30
44/44 - 0s - 7ms/step - accuracy: 0.8653 - loss: 0.3879 - val_accuracy: 0.8553 - val_loss: 0.3785
Epoch 23/30
44/44 - 0s - 7ms/step - accuracy: 0.8682 - loss: 0.3853 - val_accuracy: 0.8553 - val_loss: 0.3753
Epoch 24/30
44/44 - 0s - 4ms/step - accuracy: 0.8682 - loss: 0.3790 - val_accuracy: 0.8553 - val_loss: 0.3690
Epoch 25/30
44/44 - 0s - 9ms/step - accuracy: 0.8768 - loss: 0.3758 - val_accuracy: 0.8684 - val_loss: 0.3658
Epoch 26/30
44/44 - 1s - 15ms/step - accuracy: 0.8797 - loss: 0.3713 - val_accuracy: 0.8684 - val_loss: 0.3626
Epoch 27/30
44/44 - 1s - 12ms/step - accuracy: 0.8797 - loss: 0.3676 - val_accuracy: 0.8816 - val_loss: 0.3592
Epoch 28/30
44/44 - 0s - 9ms/step - accuracy: 0.8797 - loss: 0.3637 - val_accuracy: 0.8816 - val_loss: 0.3574
Epoch 29/30
44/44 - 0s - 7ms/step - accuracy: 0.8797 - loss: 0.3600 - val_accuracy: 0.8947 - val_loss: 0.3535
Epoch 30/30
44/44 - 0s - 4ms/step - accuracy: 0.8797 - loss: 0.3566 - val_accuracy: 0.8947 - val_loss: 0.3512
```



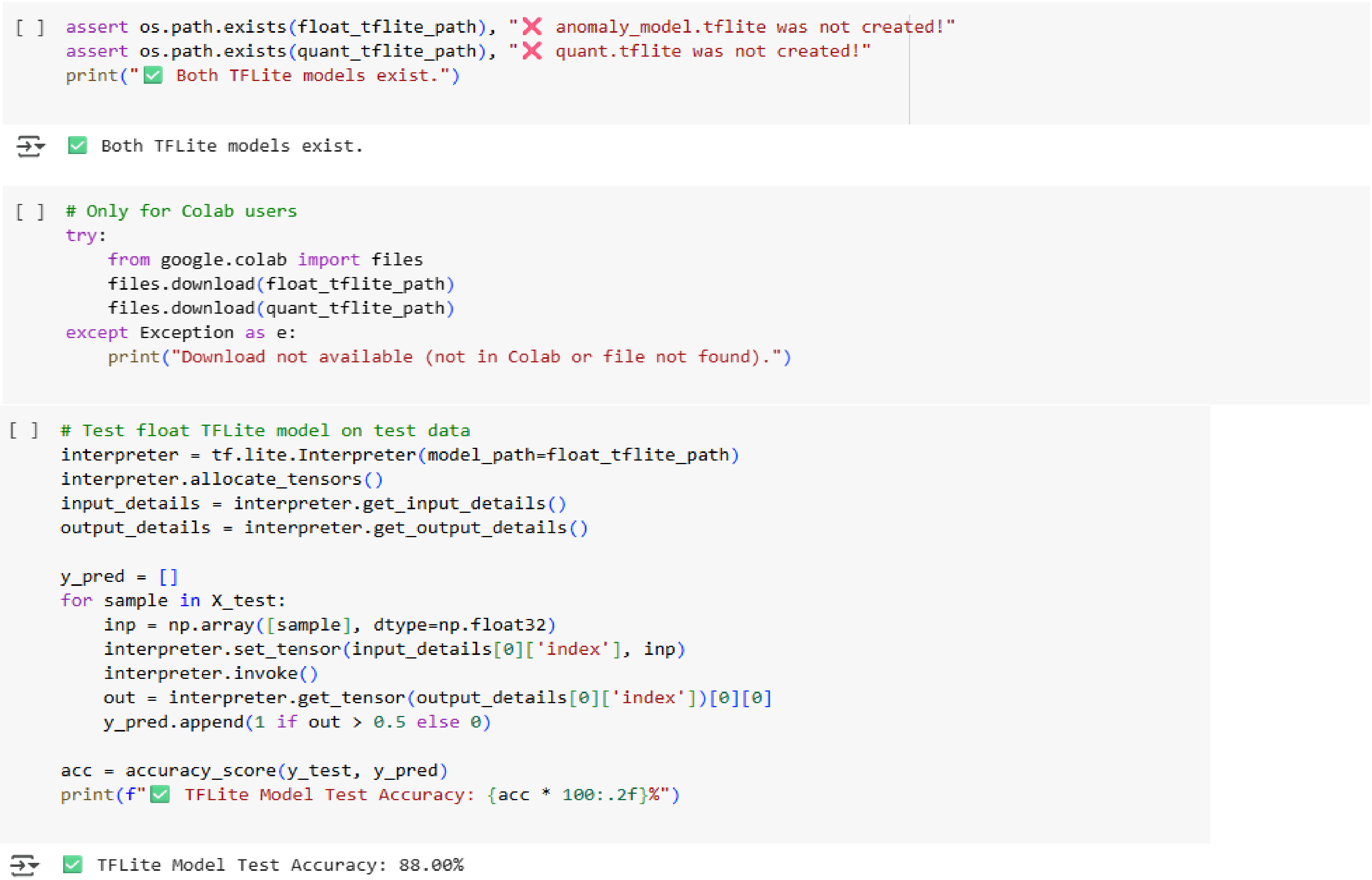
```
[ ] # Evaluate on validation and test sets
    val_loss, val_acc = model.evaluate(X_val, y_val, verbose=0)
    print(f"\n✅ Validation Accuracy: {val_acc * 100:.2f}%")
    test_loss, test_acc = model.evaluate(X_test, y_test, verbose=0)
    print(f"✅ Test Accuracy: {test_acc * 100:.2f}%")
```

```
✅ Validation Accuracy: 89.47%
✅ Test Accuracy: 88.00%
```

```
[ ] # Save the trained Keras model
    model.save("base_model.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recomme
```

```
# FLOAT TFLite model
float_tflite_path = "anomaly_model.tflite"
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
with open(float_tflite_path, "wb") as f:
    f.write(tflite_model)
print(f"✅ Saved float TFLite model as: {float_tflite_path}")

# INT8 Quantized TFLite model
quant_tflite_path = "quant.tflite"
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
def rep_data_gen():
    for sample in X_train.astype(np.float32):
        yield [sample.reshape(1, 2)]
converter.representative_dataset = rep_data_gen
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
converter.inference_input_type = tf.int8
converter.inference_output_type = tf.int8
quant_model = converter.convert()
with open(quant_tflite_path, "wb") as f:
    f.write(quant_model)
print(f"✅ Saved quantized TFLite model as: {quant_tflite_path}")
```



```python
assert os.path.exists(float_tflite_path), "❌ anomaly_model.tflite was not created!"
assert os.path.exists(quant_tflite_path), "❌ quant.tflite was not created!"
print("✅ Both TFLite models exist.")
```

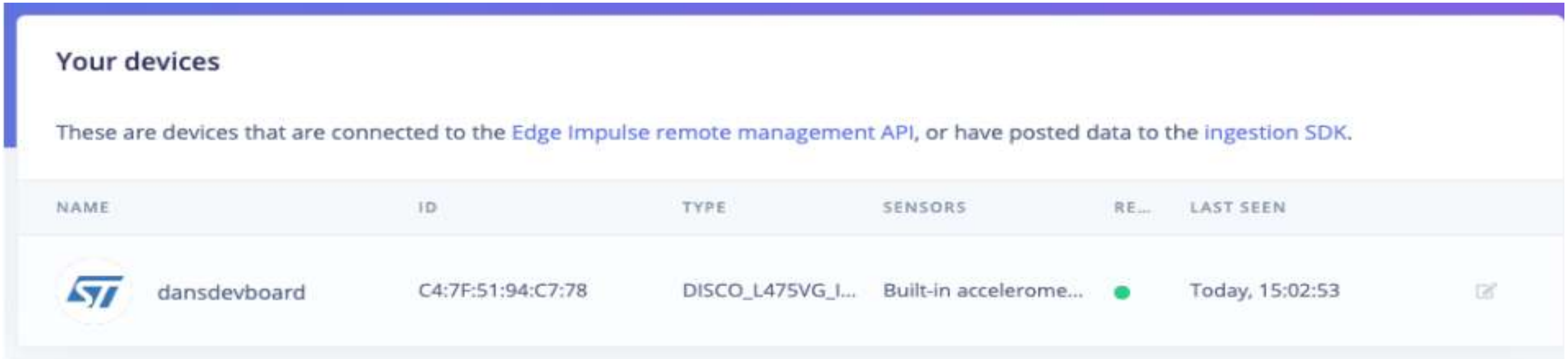
```
✅ Both TFLite models exist.
```

```python
# Only for Colab users
try:
    from google.colab import files
    files.download(float_tflite_path)
    files.download(quant_tflite_path)
except Exception as e:
    print("Download not available (not in Colab or file not found).")
```

```python
# Test float TFLite model on test data
interpreter = tf.lite.Interpreter(model_path=float_tflite_path)
interpreter.allocate_tensors()
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

y_pred = []
for sample in X_test:
    inp = np.array([sample], dtype=np.float32)
    interpreter.set_tensor(input_details[0]['index'], inp)
    interpreter.invoke()
    out = interpreter.get_tensor(output_details[0]['index'])[0][0]
    y_pred.append(1 if out > 0.5 else 0)

acc = accuracy_score(y_test, y_pred)
print(f"✅ TFLite Model Test Accuracy: {acc * 100:.2f}%")
```

```
✅ TFLite Model Test Accuracy: 88.00%
```

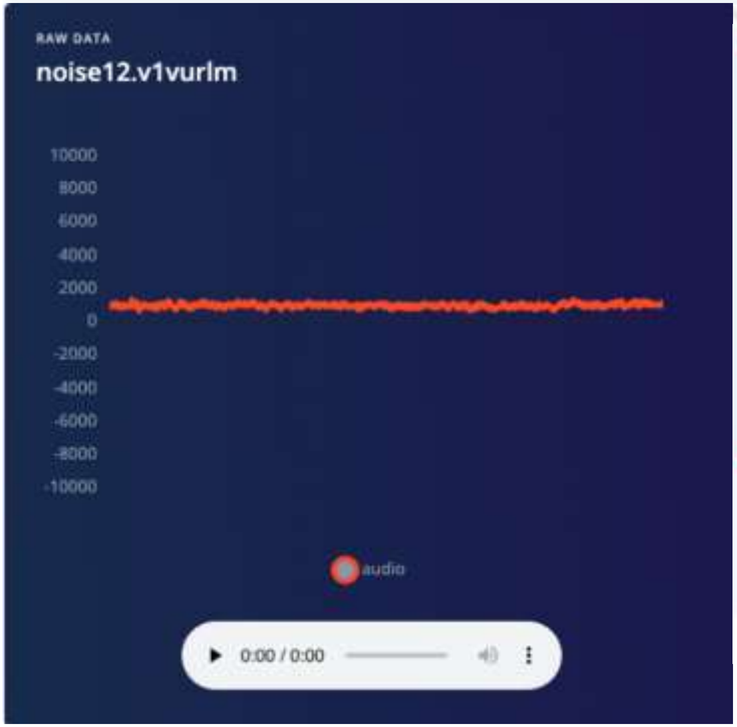
Edge Impulse Features

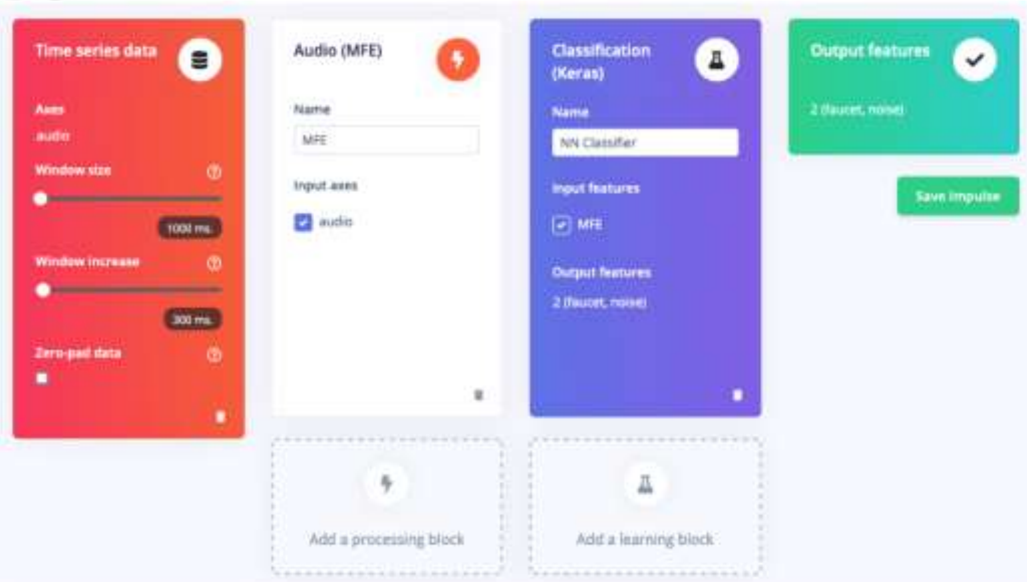


Your devices

These are devices that are connected to the Edge Impulse remote management API, or have posted data to the ingestion SDK.

NAME	ID	TYPE	SENSORS	RE...	LAST SEEN
dansdevboard	C4:7F:51:94:C7:78	DISCO_L475VG_I...	Built-in accelerome...	●	Today, 15:02:53





Time series data

Axes
audio

Window size
1000 ms.

Window increase
300 ms.

Zero-pad data
□

Audio (MFE)

Name
MFE

Input axes
☑ audio

Classification (Keras)

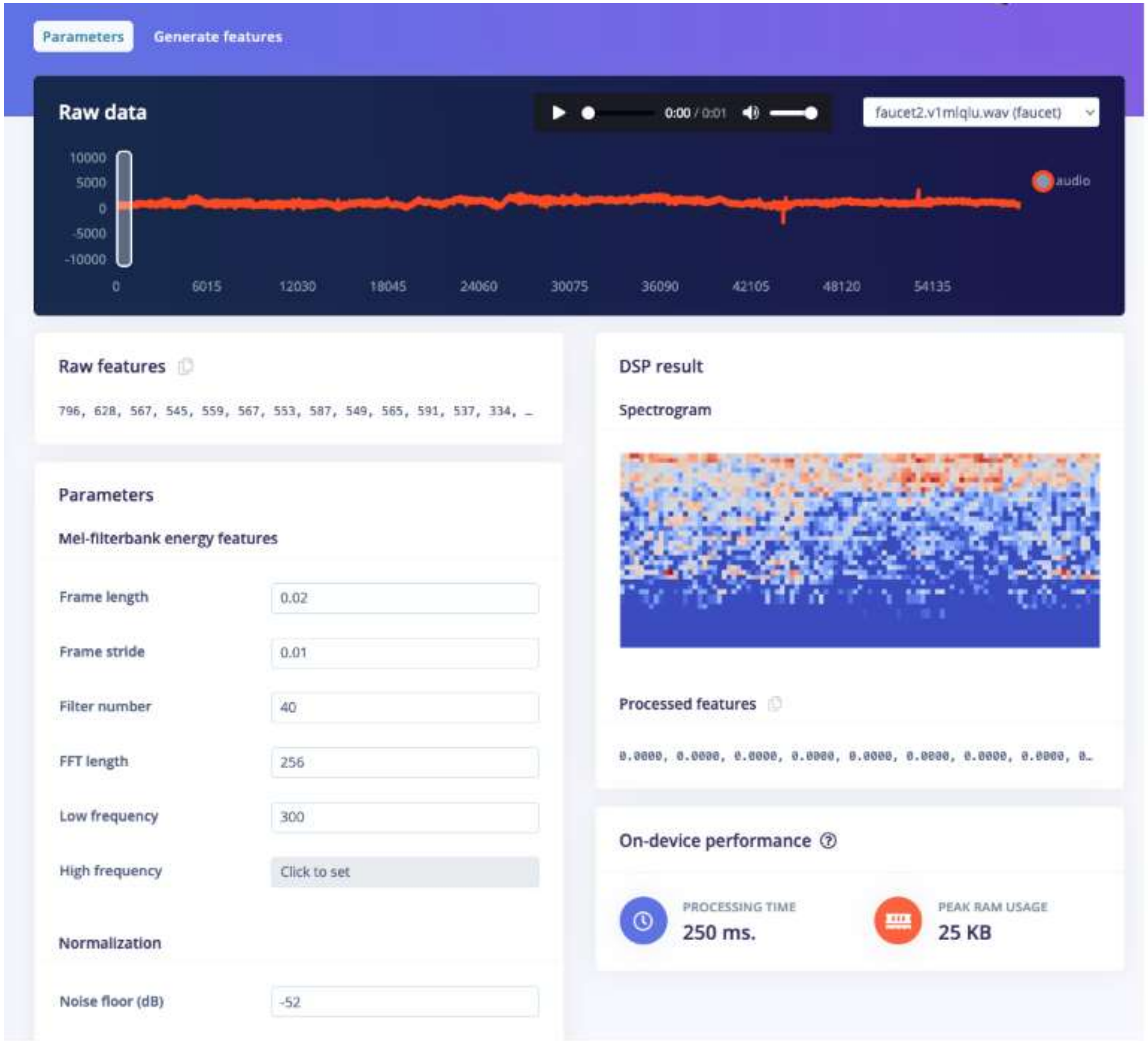
Name
NN Classifier

Input features
☑ MFE

Output features
2 (faucet, noise)

Output features
2 (faucet, noise)

The impulse, with one processing block and one learning block.



Raw data

Raw features

```
796, 628, 567, 545, 559, 567, 553, 587, 549, 565, 591, 537, 334, …
```

Parameters

Mel-filterbank energy features

Frame length: 0.02

Frame stride: 0.01

Filter number: 40

FFT length: 256

Low frequency: 300

High frequency: Click to set

Normalization

Noise floor (dB): -52

DSP result

Spectrogram

Processed features

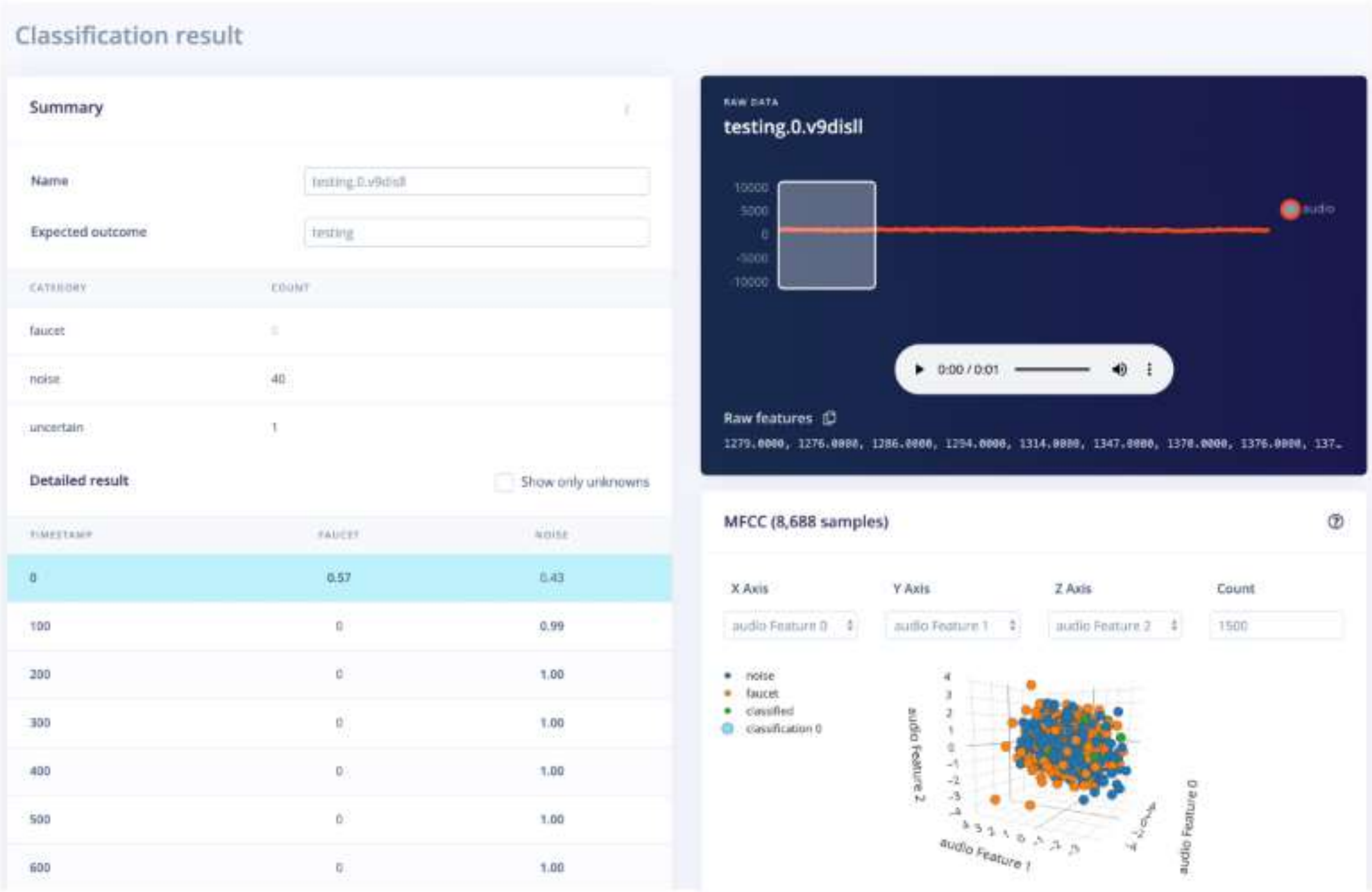
```
0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0…
```

On-device performance ⓘ

PROCESSING TIME: 250 ms.

PEAK RAM USAGE: 25 KB

The MFE page.



Classification result

Summary

Name: testing.0.v9disll

Expected outcome: testing

CATEGORY	COUNT
faucet	0
noise	40
uncertain	1

Detailed result

□ Show only unknowns

TIMESTAMP	FAUCET	NOISE
0	0.57	0.43
100	0	0.99
200	0	1.00
300	0	1.00
400	0	1.00
500	0	1.00
600	0	1.00

RAW DATA

testing.0.v9disll

Raw features

```
1279.0000, 1276.0000, 1286.0000, 1294.0000, 1314.0000, 1347.0000, 1370.0000, 1376.0000, 137…
```

MFCC (8,688 samples)

X Axis: audio Feature 0

Y Axis: audio Feature 1

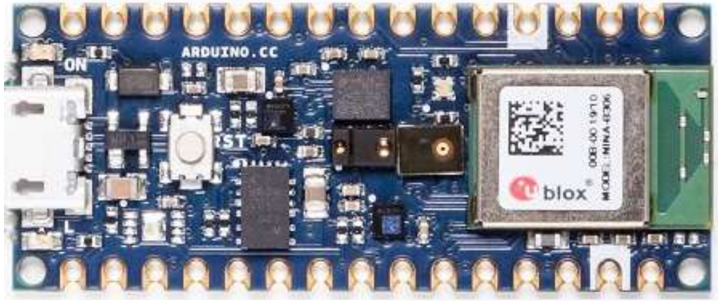
Z Axis: audio Feature 2

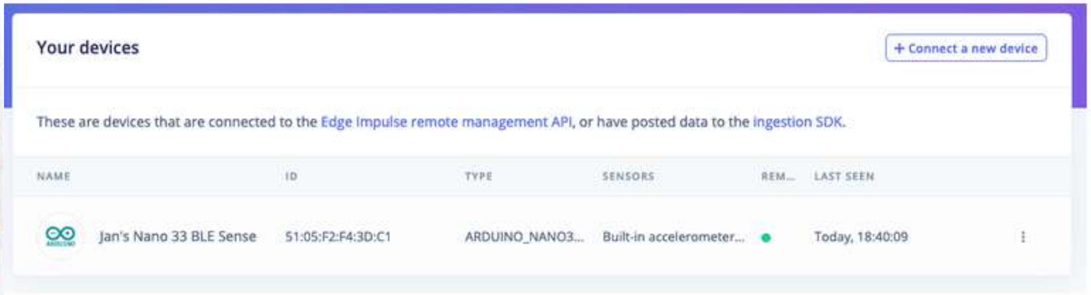
Count: 1500

The results of classifying a new sample.

Edge Impulse Doc (Embedded Device)

Arduino Nano 33 BLE Sense

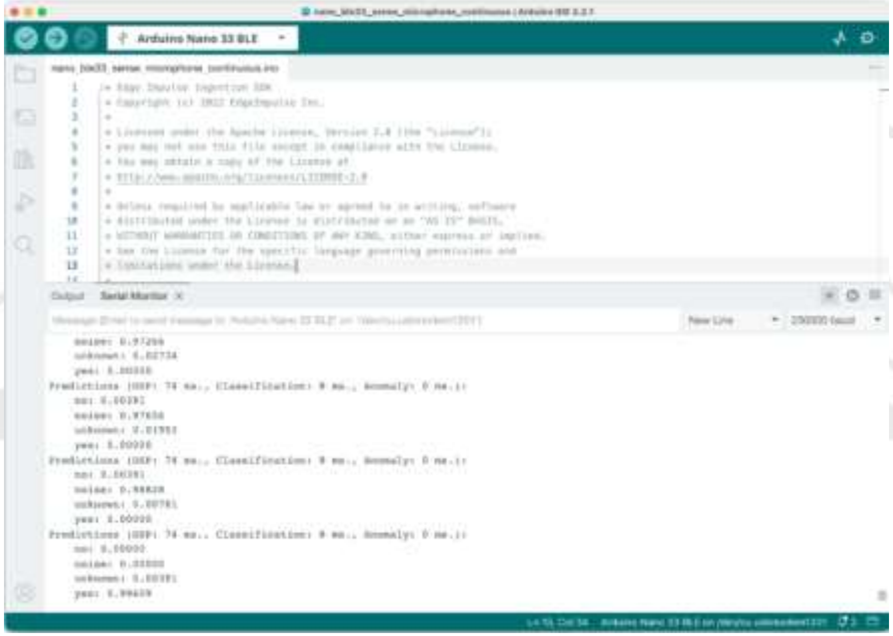




Your devices

These are devices that are connected to the Edge Impulse remote management API, or have posted data to the ingestion SDK.

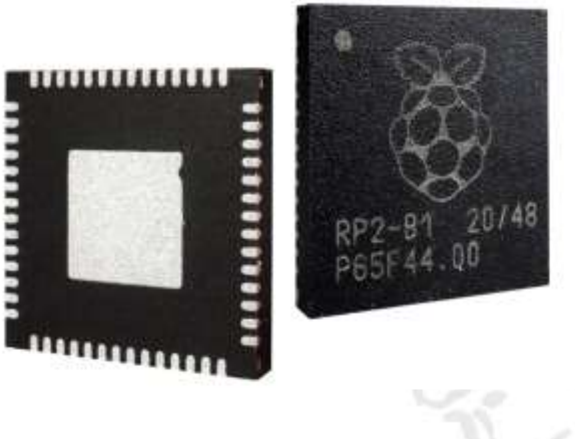
NAME	ID	TYPE	SENSORS	REM...	LAST SEEN
Jan's Nano 33 BLE Sense	51:05:F2:F4:3D:C1	ARDUINO_NANO3...	Built-in accelerometer...	●	Today, 18:40:09


nano_ble33_sense_microphone_continuous | Arduino IDE 2.2.1
Arduino Nano 33 BLE
nano_ble33_sense_microphone_continuous.ino
Output
Serial Monitor
New Line
230400 baud

```
/* Edge Impulse ingestion SDK
 * Copyright (c) 2022 EdgeImpulse Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
```

```
    noise: 0.97266
    unknown: 0.02734
    yes: 0.00000
Predictions (DSP: 74 ms., Classification: 8 ms., Anomaly: 0 ms.):
    no: 0.00391
    noise: 0.97656
    unknown: 0.01953
    yes: 0.00000
Predictions (DSP: 74 ms., Classification: 8 ms., Anomaly: 0 ms.):
    no: 0.00391
    noise: 0.98828
    unknown: 0.00781
    yes: 0.00000
Predictions (DSP: 74 ms., Classification: 8 ms., Anomaly: 0 ms.):
    no: 0.00000
    noise: 0.00000
    unknown: 0.00391
    yes: 0.99609
```

Raspberry Pi RP2350

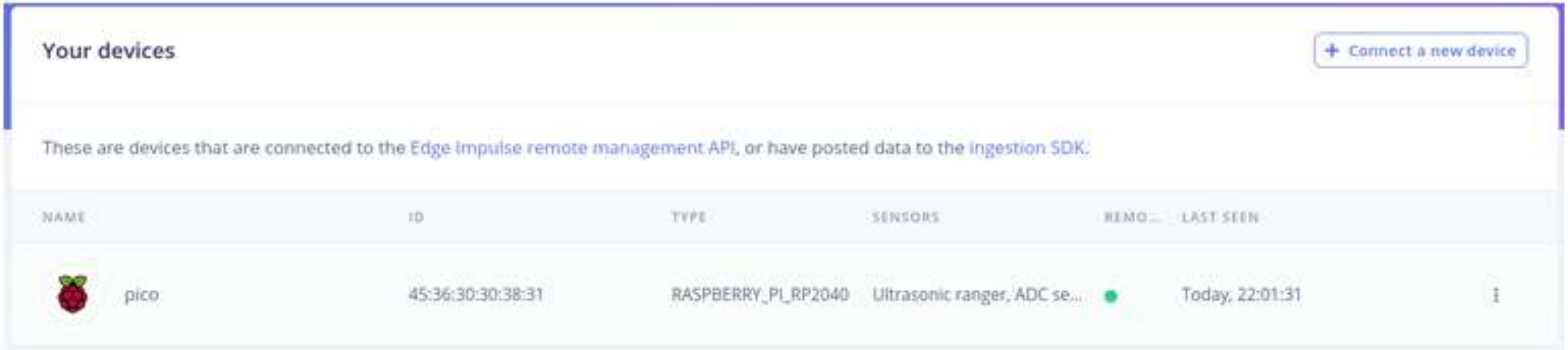


Using with other RP2040 boards

While RP2040 is a relatively new microcontroller, it was already utilized to build several boards:

- The official Raspberry Pi Pico RP2040
- Arducam Pico4ML (Camera, screen and microphone)
- Seeed Studio XIAO RP2040 (extremely small footprint)
- Black Adafruit Feather RP2040 (built-in LiPoly charger)

And others. While pre-built Edge Impulse firmware is mainly tested with Pico board, it is compatible with other boards, with the exception of I2C sensors and microphone - different boards use different pins for peripherals, so if you'd like to use LSM6DS3/LSM6DSOX accelerometer & gyroscope modules or microphone, you will need to change pin values in Edge Impulse RP2040 firmware source code, recompile it and upload it to the board.



Your devices

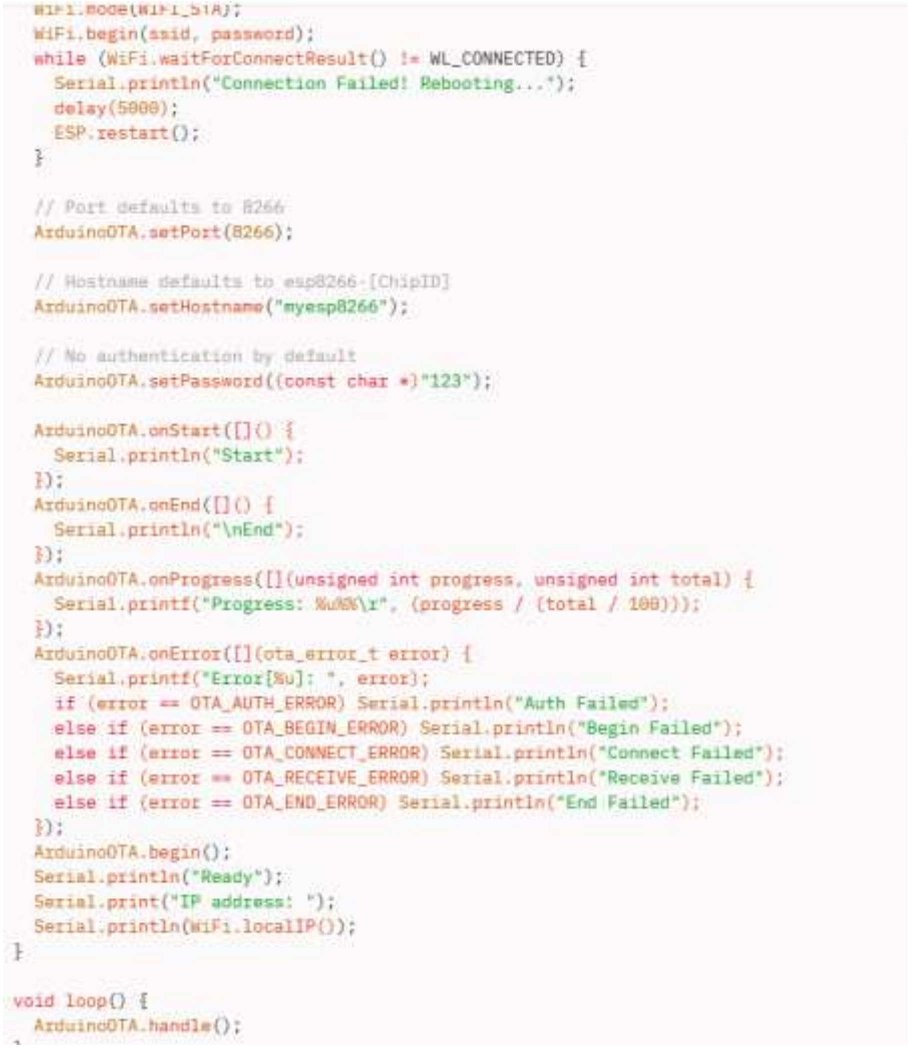
These are devices that are connected to the Edge Impulse remote management API, or have posted data to the ingestion SDK.

NAME	ID	TYPE	SENSORS	REMO...	LAST SEEN
pico	45:36:30:30:38:31	RASPBERRY_PI_RP2040	Ultrasonic ranger, ADC se...	●	Today, 22:01:31

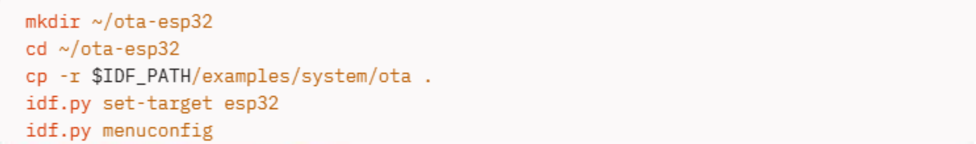
Raspberry Pi Pico board connected to Edge Impulse Studio.

ESP8266 Board

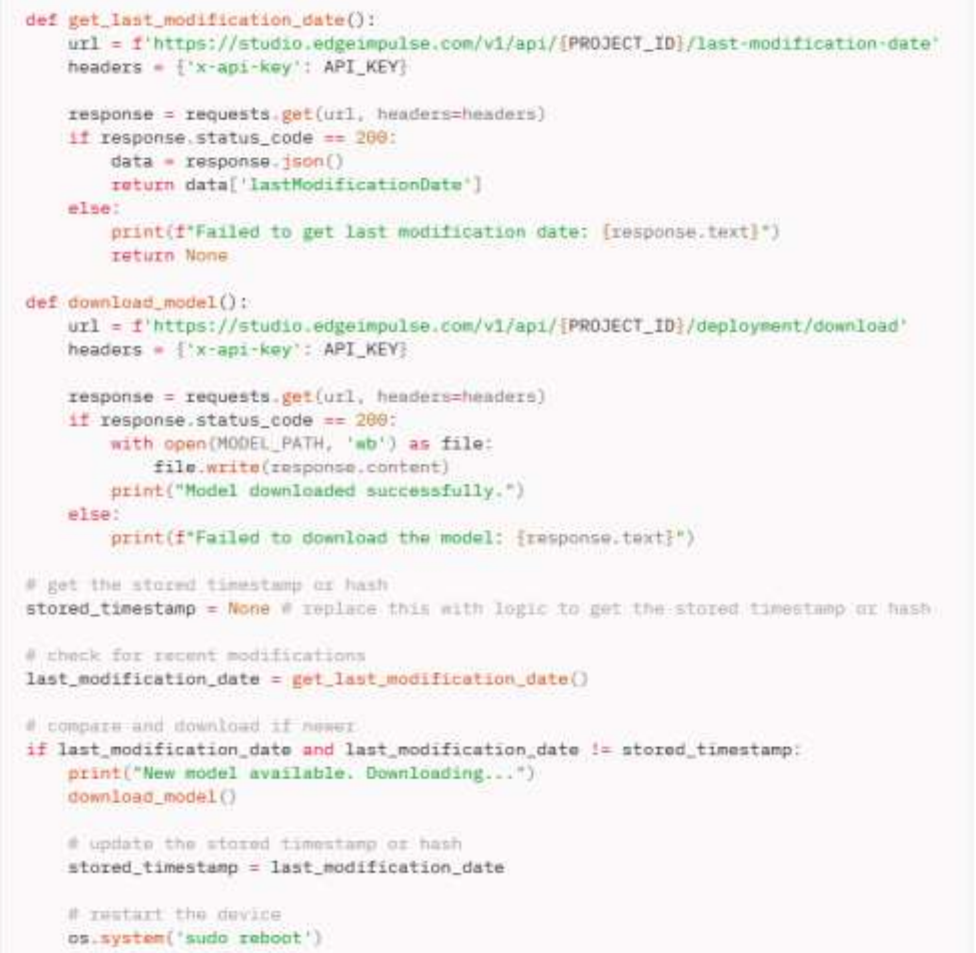
with Arduino IDE (for ESP32)



```
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("Connection Failed! Rebooting...");
    delay(5000);
    ESP.restart();
  }

  // Port defaults to 8266
  ArduinoOTA.setPort(8266);

  // Hostname defaults to esp8266-[ChipID]
  ArduinoOTA.setHostname("myesp8266");

  // No authentication by default
  ArduinoOTA.setPassword((const char *)"123");

  ArduinoOTA.onStart([]() {
    Serial.println("Start");
  });
  ArduinoOTA.onEnd([]() {
    Serial.println("\nEnd");
  });
  ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
  });
  ArduinoOTA.onError([](ota_error_t error) {
    Serial.printf("Error[%u]: ", error);
    if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
    else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
    else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
    else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
    else if (error == OTA_END_ERROR) Serial.println("End Failed");
  });
  ArduinoOTA.begin();
  Serial.println("Ready");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  ArduinoOTA.handle();
```



```
mkdir ~/ota-esp32
cd ~/ota-esp32
cp -r $IDF_PATH/examples/system/ota .
idf.py set-target esp32
idf.py menuconfig
```

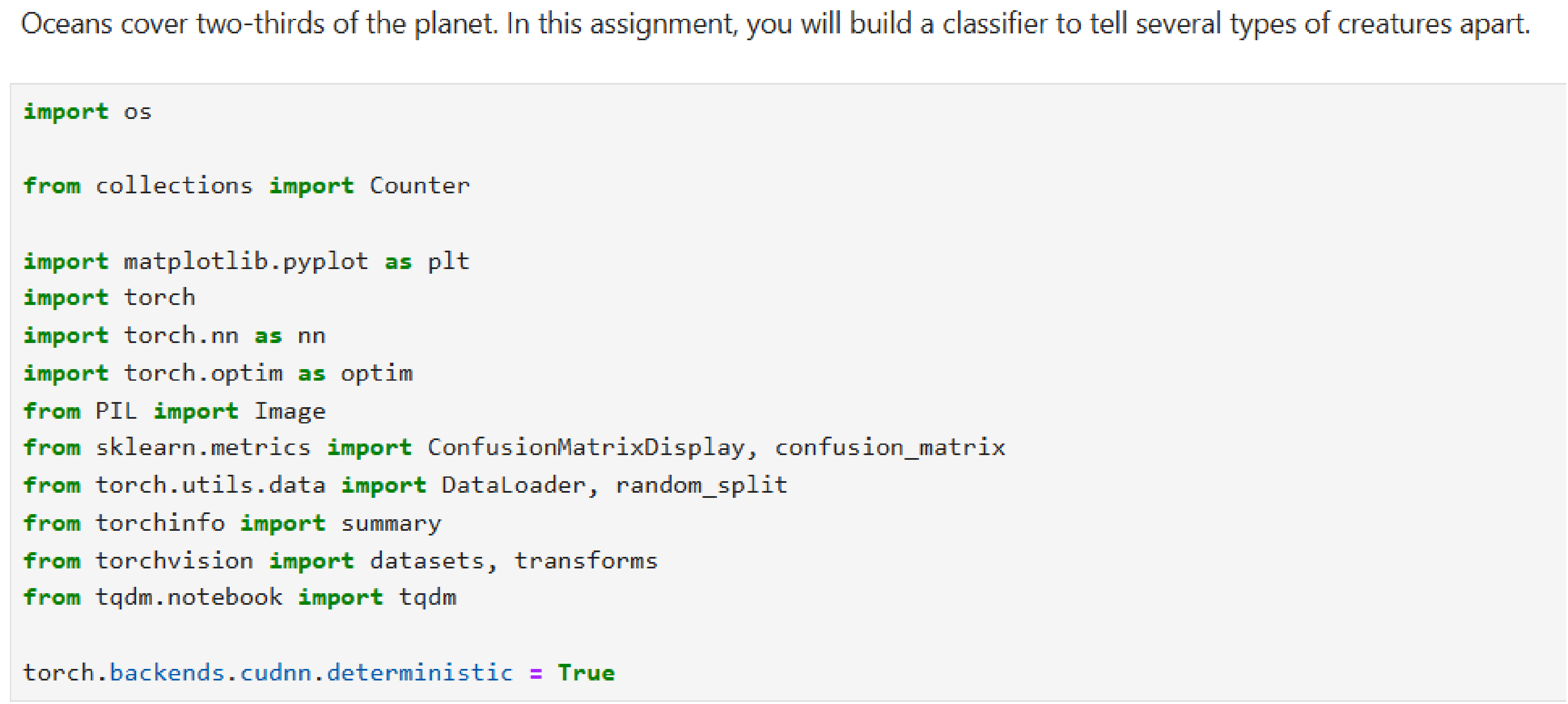


```
def get_last_modification_date():
    url = f'https://studio.edgeimpulse.com/v1/api/{PROJECT_ID}/last-modification-date'
    headers = {'x-api-key': API_KEY}

    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        data = response.json()
        return data['lastModificationDate']
    else:
        print(f"Failed to get last modification date: {response.text}")
        return None

def download_model():
    url = f'https://studio.edgeimpulse.com/v1/api/{PROJECT_ID}/deployment/download'
    headers = {'x-api-key': API_KEY}

    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        with open(MODEL_PATH, 'wb') as file:
            file.write(response.content)
        print("Model downloaded successfully.")
    else:
        print(f"Failed to download the model: {response.text}")

# get the stored timestamp or hash
stored_timestamp = None # replace this with logic to get the stored timestamp or hash

# check for recent modifications
last_modification_date = get_last_modification_date()

# compare and download if newer
if last_modification_date and last_modification_date != stored_timestamp:
    print("New model available. Downloading...")
    download_model()

    # update the stored timestamp or hash
    stored_timestamp = last_modification_date

    # restart the device
    os.system('sudo reboot')
```

Multi-Classification Model

Steps Involved:-

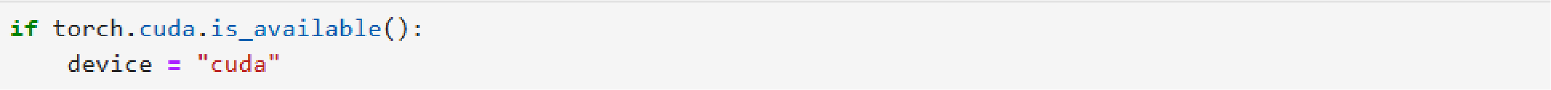
Libraries Installation:-



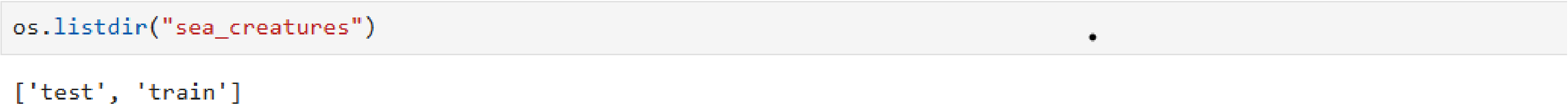
Oceans cover two-thirds of the planet. In this assignment, you will build a classifier to tell several types of creatures apart.

```
import os

from collections import Counter

import matplotlib.pyplot as plt
import torch
import torch.nn as nn
import torch.optim as optim
from PIL import Image
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
from torch.utils.data import DataLoader, random_split
from torchinfo import summary
from torchvision import datasets, transforms
from tqdm.notebook import tqdm

torch.backends.cudnn.deterministic = True
```

Environment Setup:-

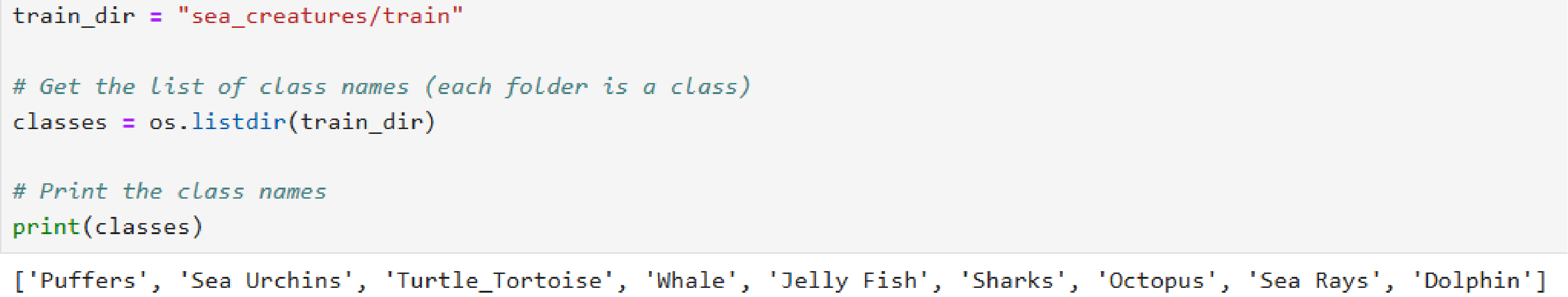


```
if torch.cuda.is_available():
    device = "cuda"
```



```
os.listdir("sea_creatures")
```

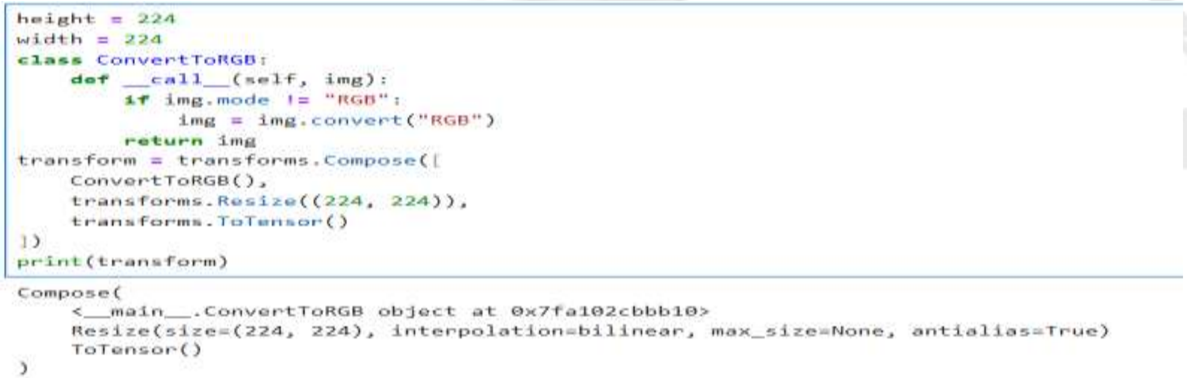
```
['test', 'train']
```



```
train_dir = "sea_creatures/train"

# Get the list of class names (each folder is a class)
classes = os.listdir(train_dir)

# Print the class names
print(classes)
```

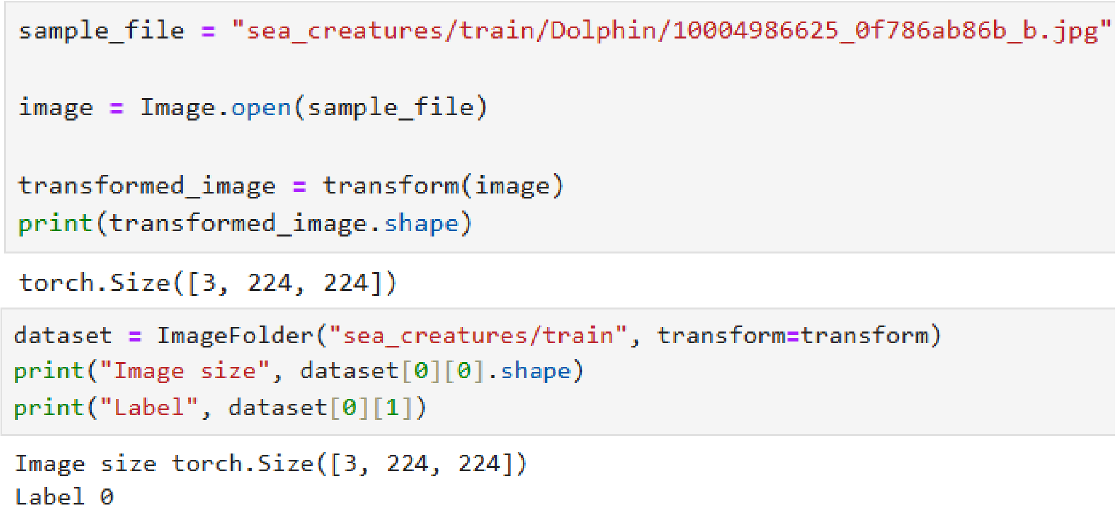
```
['Puffers', 'Sea Urchins', 'Turtle_Tortoise', 'Whale', 'Jelly Fish', 'Sharks', 'Octopus', 'Sea Rays', 'Dolphin']
```

Transform Pipeline:-



```
height = 224
width = 224
class ConvertToRGB:
    def __call__(self, img):
        if img.mode != "RGB":
            img = img.convert("RGB")
        return img
transform = transforms.Compose([
    ConvertToRGB(),
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])
print(transform)
```

```
Compose(
    <__main__.ConvertToRGB object at 0x7fa102cbbb10>
    Resize(size=(224, 224), interpolation=bilinear, max_size=None, antialias=True)
    ToTensor()
)
```



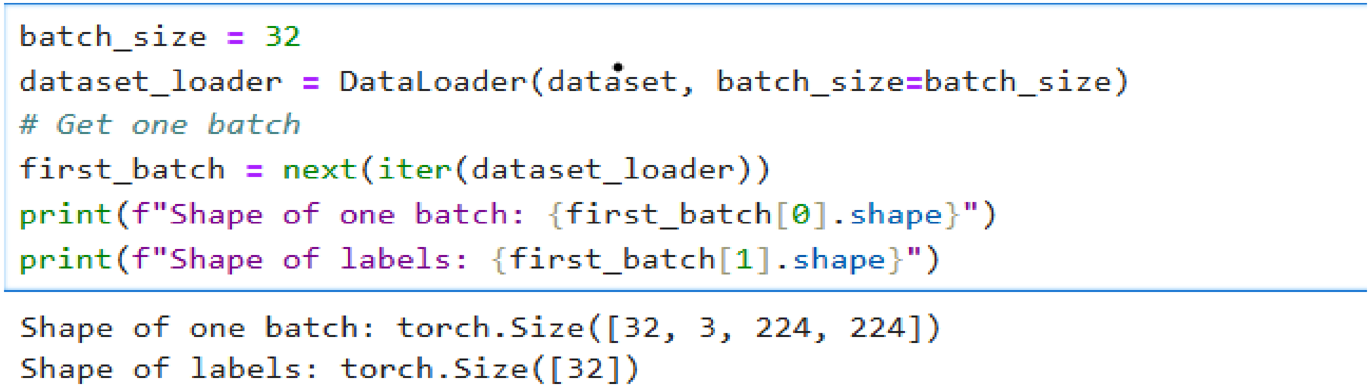
```
sample_file = "sea_creatures/train/Dolphin/10004986625_0f786ab86b_b.jpg"

image = Image.open(sample_file)

transformed_image = transform(image)
print(transformed_image.shape)
```

```
torch.Size([3, 224, 224])
```

```
dataset = ImageFolder("sea_creatures/train", transform=transform)
print("Image size", dataset[0][0].shape)
print("Label", dataset[0][1])
```

```
Image size torch.Size([3, 224, 224])
Label 0
```

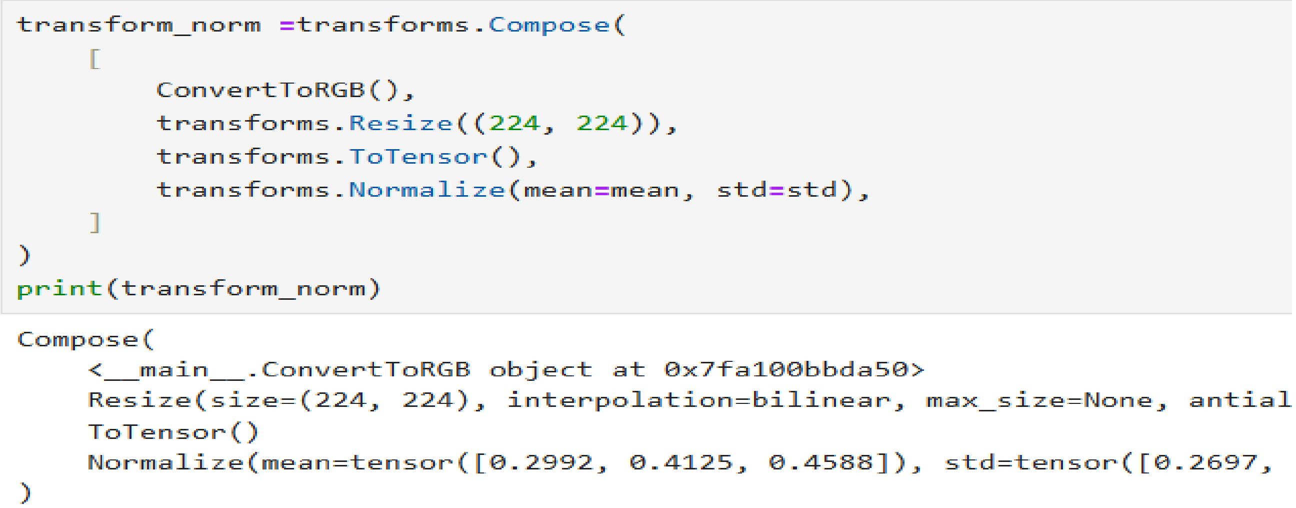
Data loader(batch size-30)



```
batch_size = 32
dataset_loader = DataLoader(dataset, batch_size=batch_size)
# Get one batch
first_batch = next(iter(dataset_loader))
print(f"Shape of one batch: {first_batch[0].shape}")
print(f"Shape of labels: {first_batch[1].shape}")
```

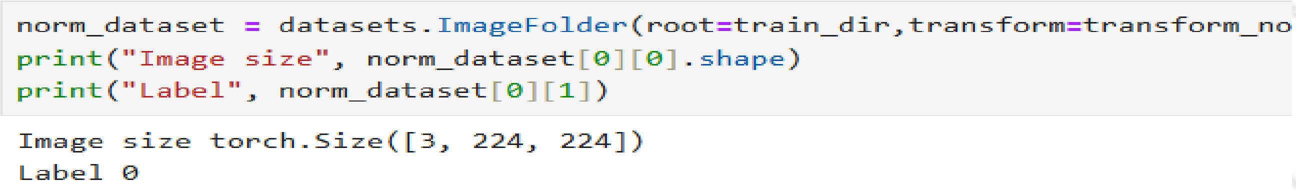
```
Shape of one batch: torch.Size([32, 3, 224, 224])
Shape of labels: torch.Size([32])
```

Transform Normalize:-



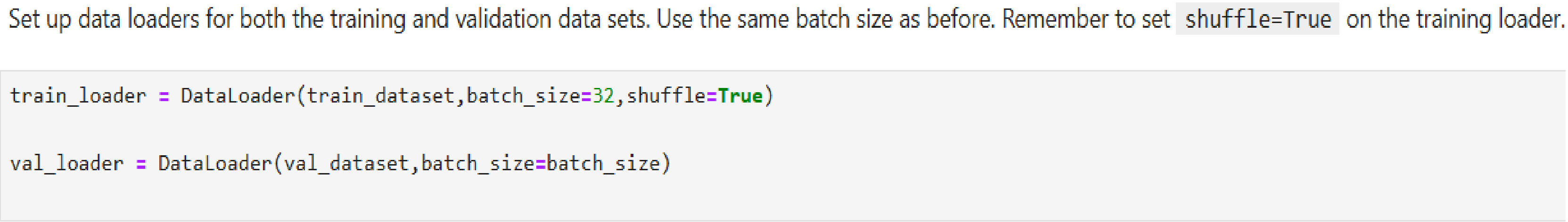
```
transform_norm =transforms.Compose(
    [
        ConvertToRGB(),
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(mean=mean, std=std),
    ]
)
print(transform_norm)
```

```
Compose(
    <__main__.ConvertToRGB object at 0x7fa100bbda50>
    Resize(size=(224, 224), interpolation=bilinear, max_size=None, antial
    ToTensor()
    Normalize(mean=tensor([0.2992, 0.4125, 0.4588]), std=tensor([0.2697,
)
```



```
norm_dataset = datasets.ImageFolder(root=train_dir,transform=transform_no
print("Image size", norm_dataset[0][0].shape)
print("Label", norm_dataset[0][1])
```

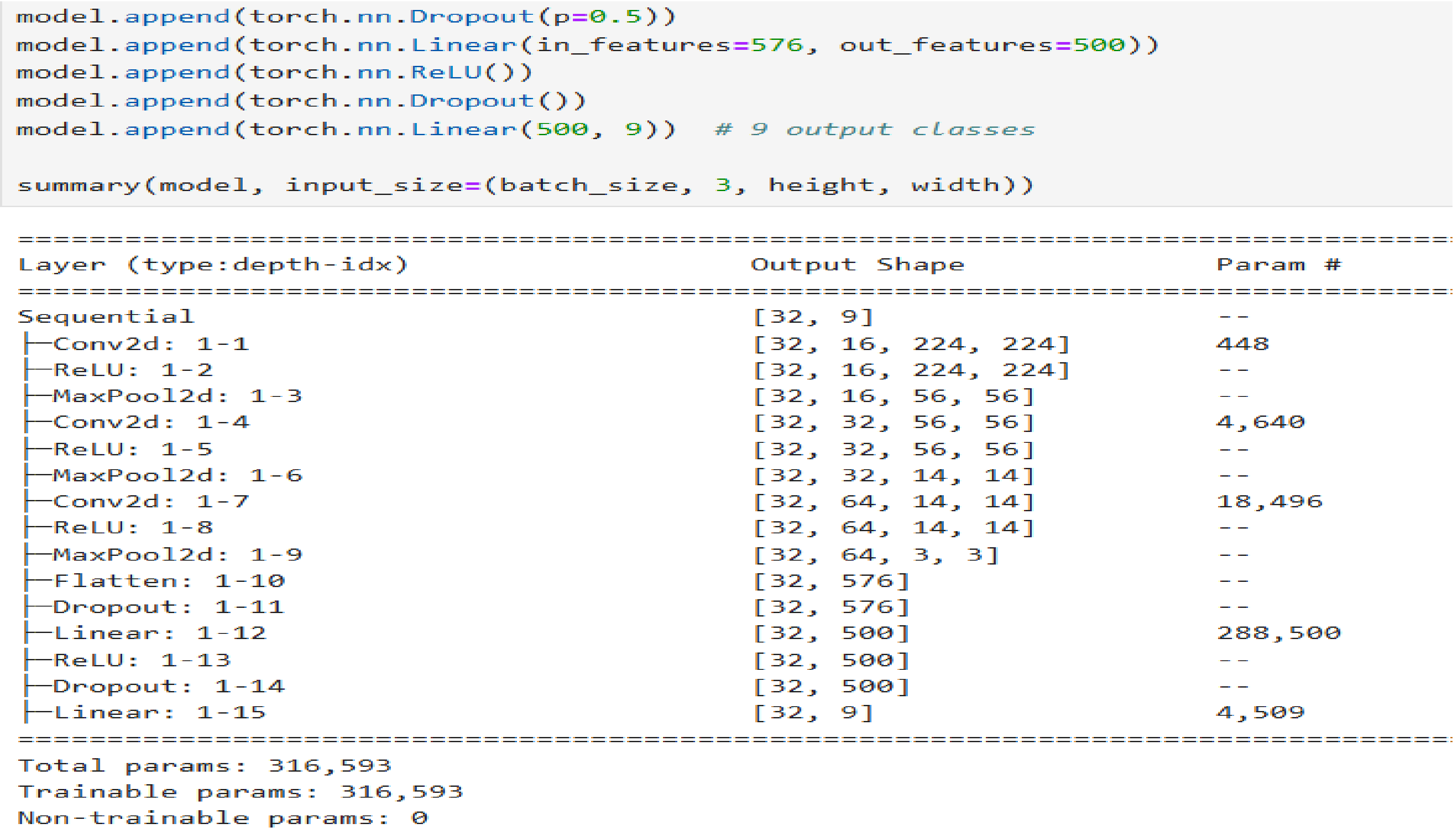
```
Image size torch.Size([3, 224, 224])
Label 0
```



Set up data loaders for both the training and validation data sets. Use the same batch size as before. Remember to set `shuffle=True` on the training loader.

```
train_loader = DataLoader(train_dataset,batch_size=32,shuffle=True)


val_loader = DataLoader(val_dataset,batch_size=batch_size)
```

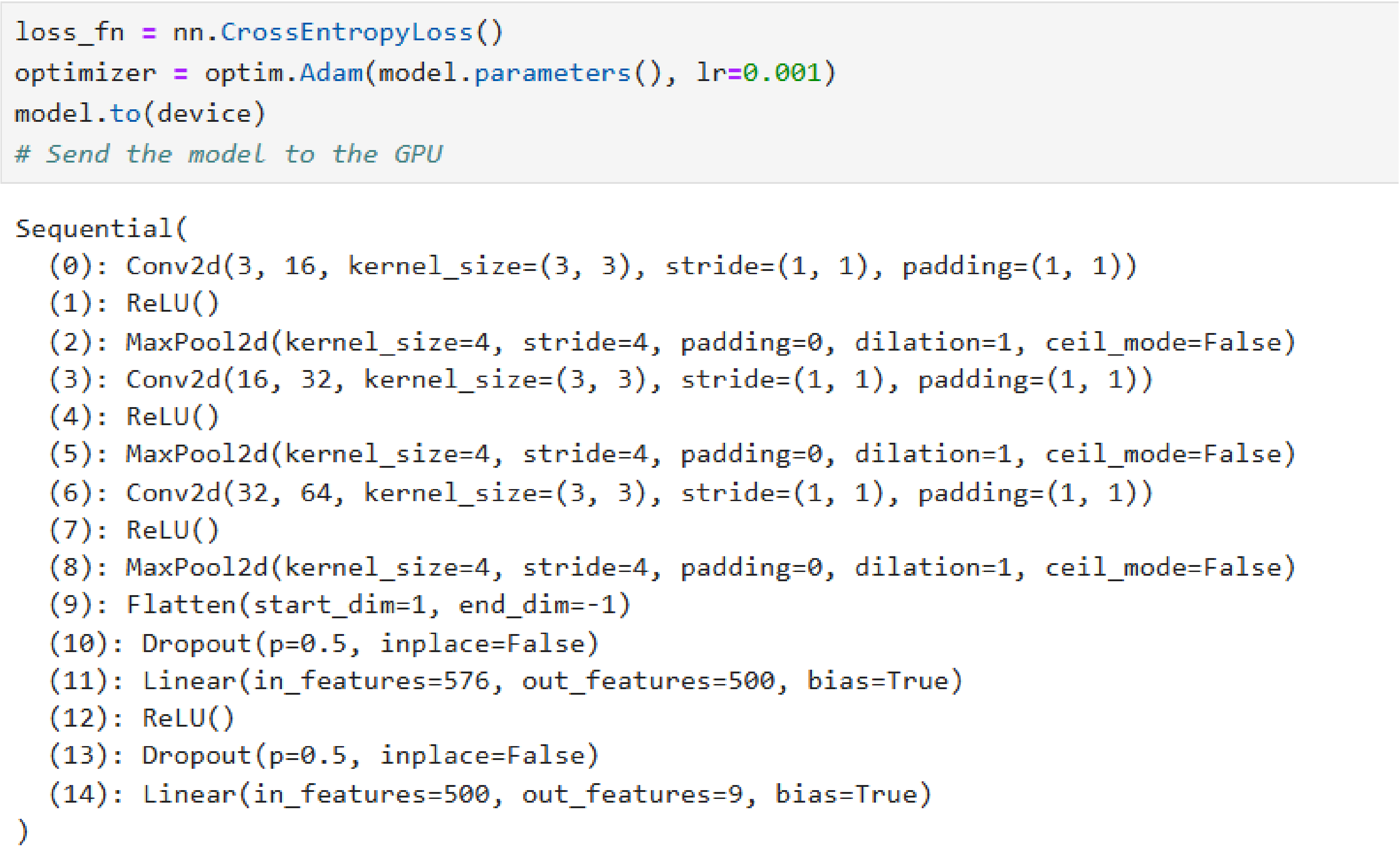
- Build Model:-



```python
model.append(torch.nn.Dropout(p=0.5))
model.append(torch.nn.Linear(in_features=576, out_features=500))
model.append(torch.nn.ReLU())
model.append(torch.nn.Dropout())
model.append(torch.nn.Linear(500, 9))  # 9 output classes

summary(model, input_size=(batch_size, 3, height, width))
```

```
================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
================================================================================
Sequential                               [32, 9]                   --
├─Conv2d: 1-1                            [32, 16, 224, 224]        448
├─ReLU: 1-2                              [32, 16, 224, 224]        --
├─MaxPool2d: 1-3                         [32, 16, 56, 56]          --
├─Conv2d: 1-4                            [32, 32, 56, 56]          4,640
├─ReLU: 1-5                              [32, 32, 56, 56]          --
├─MaxPool2d: 1-6                         [32, 32, 14, 14]          --
├─Conv2d: 1-7                            [32, 64, 14, 14]          18,496
├─ReLU: 1-8                              [32, 64, 14, 14]          --
├─MaxPool2d: 1-9                         [32, 64, 3, 3]            --
├─Flatten: 1-10                          [32, 576]                 --
├─Dropout: 1-11                          [32, 576]                 --
├─Linear: 1-12                           [32, 500]                 288,500
├─ReLU: 1-13                             [32, 500]                 --
├─Dropout: 1-14                          [32, 500]                 --
├─Linear: 1-15                           [32, 9]                   4,509
================================================================================
Total params: 316,593
Trainable params: 316,593
Non-trainable params: 0
```

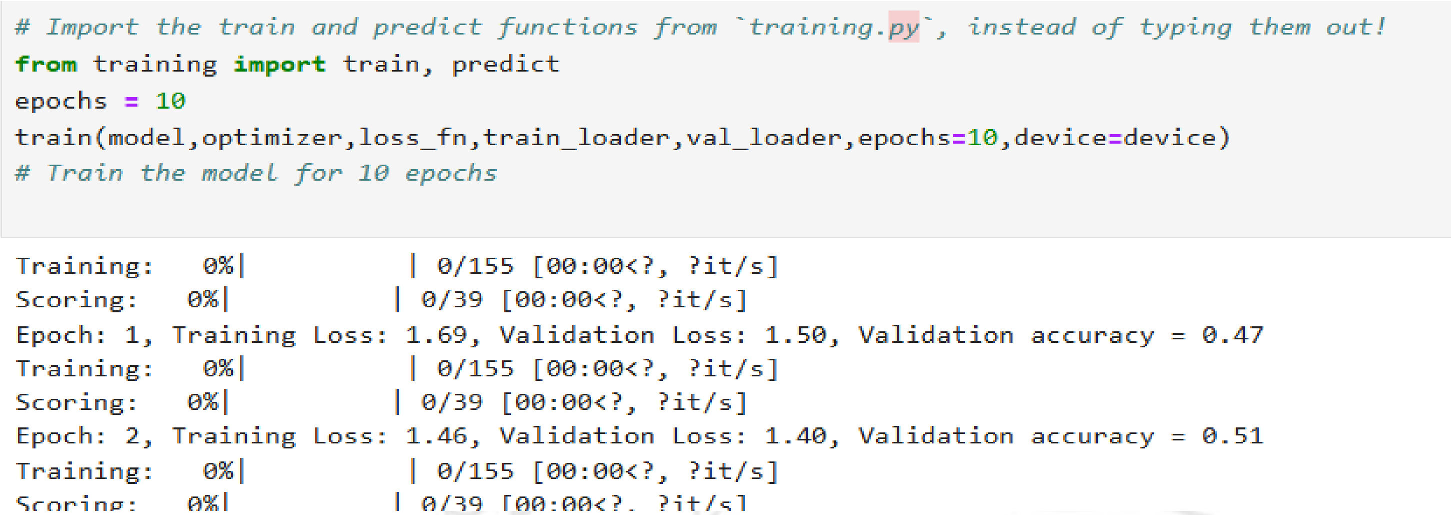
Cross Entropy Loss:-



```
loss_fn = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
model.to(device)
# Send the model to the GPU
```

```
Sequential(
  (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU()
  (2): MaxPool2d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (4): ReLU()
  (5): MaxPool2d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (6): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (7): ReLU()
  (8): MaxPool2d(kernel_size=4, stride=4, padding=0, dilation=1, ceil_mode=False)
  (9): Flatten(start_dim=1, end_dim=-1)
  (10): Dropout(p=0.5, inplace=False)
  (11): Linear(in_features=576, out_features=500, bias=True)
  (12): ReLU()
  (13): Dropout(p=0.5, inplace=False)
  (14): Linear(in_features=500, out_features=9, bias=True)
)
```

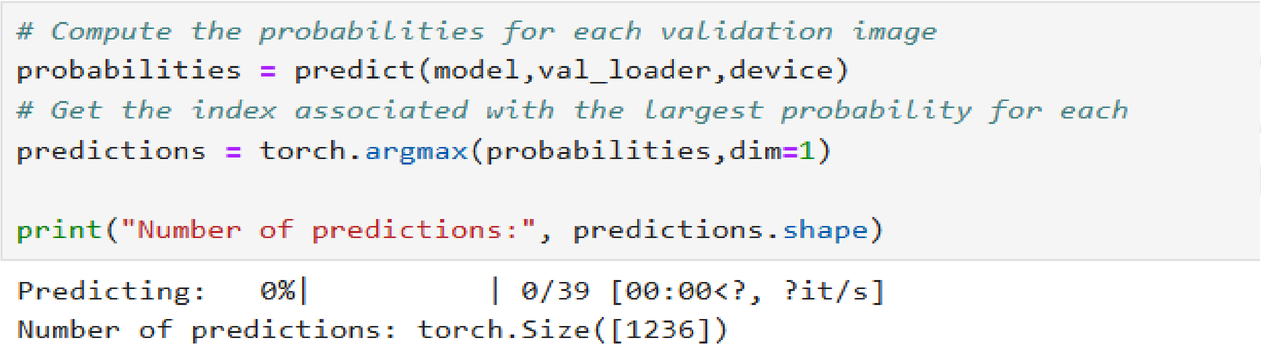
Training Of Dataset



```
# Import the train and predict functions from `training.py`, instead of typing them out!
from training import train, predict
epochs = 10
train(model,optimizer,loss_fn,train_loader,val_loader,epochs=10,device=device)
# Train the model for 10 epochs
```

```
Training:   0%|          | 0/155 [00:00<?, ?it/s]
Scoring:   0%|          | 0/39 [00:00<?, ?it/s]
Epoch: 1, Training Loss: 1.69, Validation Loss: 1.50, Validation accuracy = 0.47
Training:   0%|          | 0/155 [00:00<?, ?it/s]
Scoring:   0%|          | 0/39 [00:00<?, ?it/s]
Epoch: 2, Training Loss: 1.46, Validation Loss: 1.40, Validation accuracy = 0.51
Training:   0%|          | 0/155 [00:00<?, ?it/s]
Scoring:   0%|          | 0/39 [00:00<?, ?it/s]
```

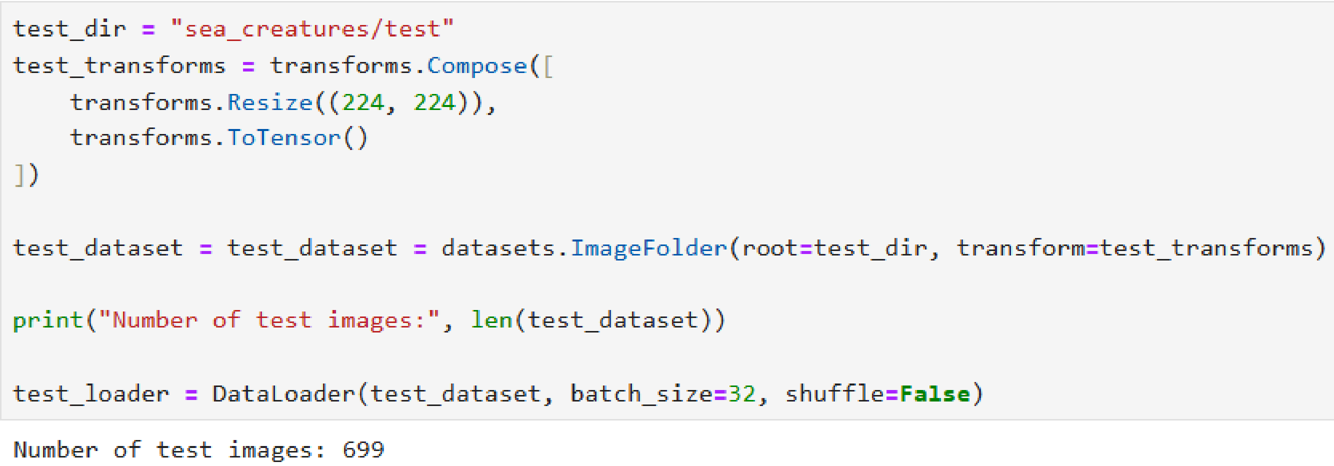
Evaluate Model Performance:-



```
# Compute the probabilities for each validation image
probabilities = predict(model,val_loader,device)
# Get the index associated with the largest probability for each
predictions = torch.argmax(probabilities,dim=1)

print("Number of predictions:", predictions.shape)
```

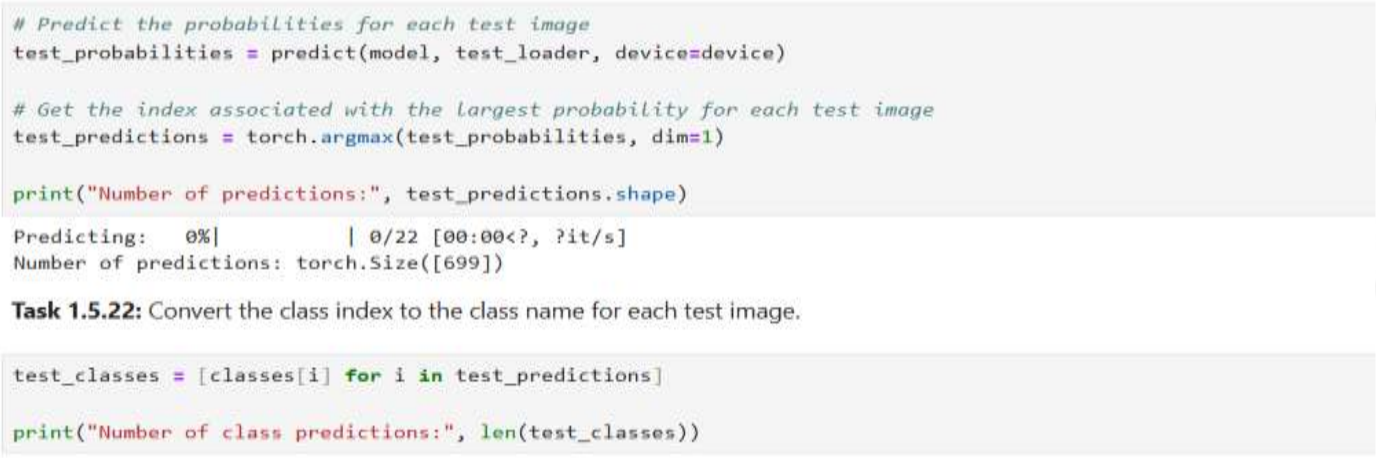
```
Predicting:   0%|          | 0/39 [00:00<?, ?it/s]
Number of predictions: torch.Size([1236])
```

Testing of dataset:-



```
test_dir = "sea_creatures/test"
test_transforms = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])

test_dataset = test_dataset = datasets.ImageFolder(root=test_dir, transform=test_transforms)

print("Number of test images:", len(test_dataset))

test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
```

```
Number of test images: 699
```



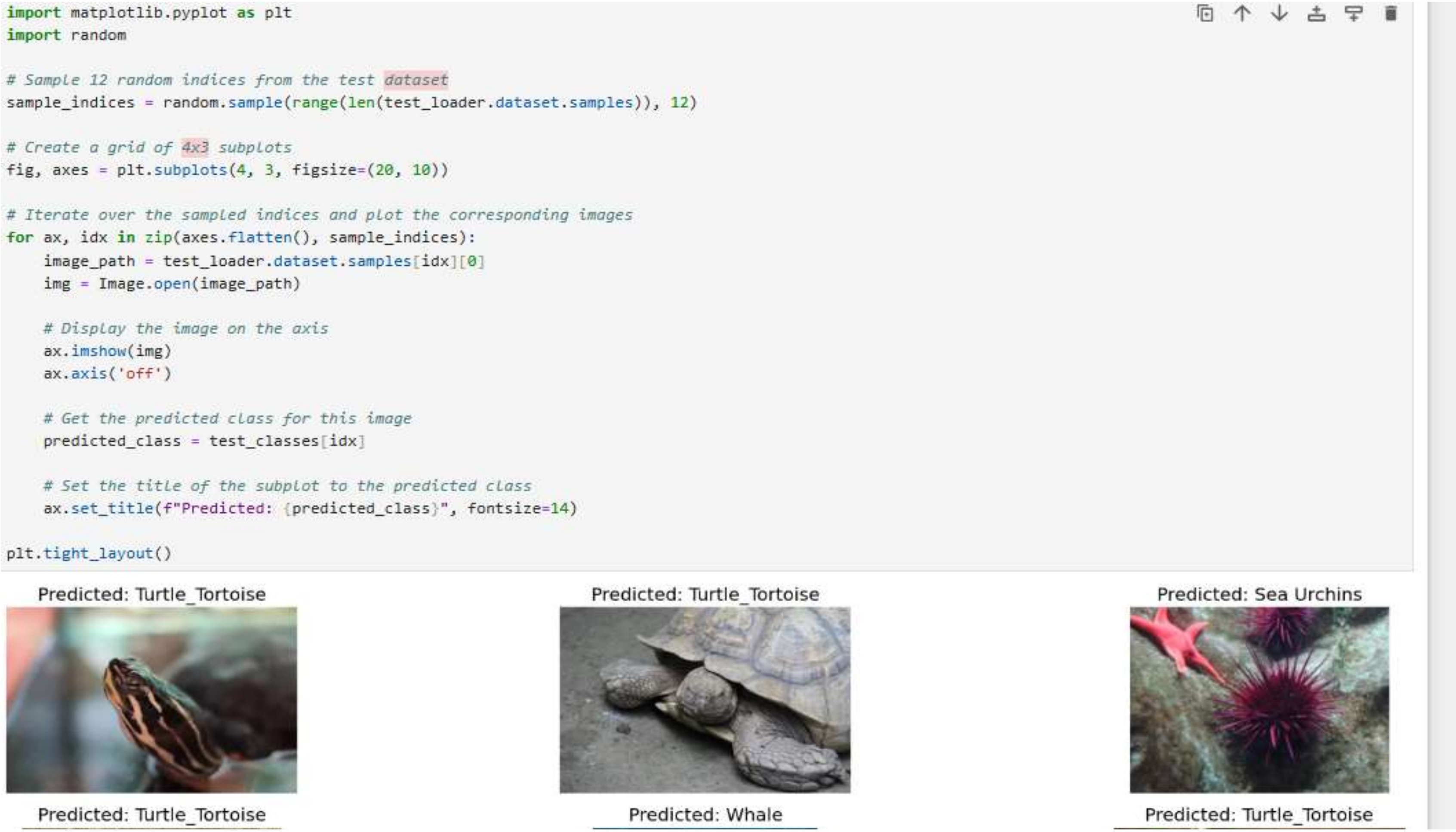
```
# Predict the probabilities for each test image
test_probabilities = predict(model, test_loader, device=device)

# Get the index associated with the largest probability for each test image
test_predictions = torch.argmax(test_probabilities, dim=1)

print("Number of predictions:", test_predictions.shape)
```

```
Predicting:   0%|          | 0/22 [00:00<?, ?it/s]
Number of predictions: torch.Size([699])
```

Task 1.5.22: Convert the class index to the class name for each test image.

```
test_classes = [classes[i] for i in test_predictions]

print("Number of class predictions:", len(test_classes))
```

Predictions through the validation dataset



```
import matplotlib.pyplot as plt
import random

# Sample 12 random indices from the test dataset
sample_indices = random.sample(range(len(test_loader.dataset.samples)), 12)

# Create a grid of 4x3 subplots
fig, axes = plt.subplots(4, 3, figsize=(20, 10))

# Iterate over the sampled indices and plot the corresponding images
for ax, idx in zip(axes.flatten(), sample_indices):
    image_path = test_loader.dataset.samples[idx][0]
    img = Image.open(image_path)

    # Display the image on the axis
    ax.imshow(img)
    ax.axis('off')

    # Get the predicted class for this image
    predicted_class = test_classes[idx]

    # Set the title of the subplot to the predicted class
    ax.set_title(f"Predicted: {predicted_class}", fontsize=14)

plt.tight_layout()
```

```
Predicted: Turtle_Tortoise        Predicted: Turtle_Tortoise        Predicted: Sea Urchins

Predicted: Turtle_Tortoise        Predicted: Whale                  Predicted: Turtle_Tortoise
```

References

- **RVfpga HarvardX edX** and **RVfpga source code**
- TinyML foundational courses and **TensorFlow Lite Micro documentation.**
- **WorldQuant University** Applied AI Lab content
- RVfpga simulator **Vibodo,Rvfpga Nexys Board,Piplines,Whisper**(Debugging and simulation) and Tracer (**GTK wave** usage), then **whisper for c program files** compilation.
- **Report** and **internal study materials.**
- **To complete for hardware Devices.**