Adebanke Damola-Fashola

ITAI 3377- AI at the Edge and IIOT Environments

Professor Patricia McManus

February 11, 2025.

Simulation Documentation

Deploying a Simple AI Model on a Simulated Edge Device using Visual Studio Code

Step 1-Set Up the Environment:

- **Install Python:** I updated the Python version on my system to the latest version 3.13.2.
- **Install VS Code:** I downloaded Visual Studio Code and installed it on my existing Anaconda platform.
- **Install TensorFlow:** I ran code 'pip install tensorflow numpy matplotlib' on the VS code terminal.
- **Install Edge Impulse CLI:** I downloaded Node.js from https://nodejs.org/en and installed it on my system. Then I ran code 'npm install -g edge-impulse-cli' on the powershell terminal.

Step 2 - Prepare the Dataset:

- **Load and Preprocess the Data:** I created a new file named deploy_mnist and selected All Files as the type. This file was used for coding in Python. I loaded the MNIST dataset.
- **Input:**

```python
import tensorflow as tf
from tensorflow.keras.datasets import mnist # type: ignore

# Load dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize the pixel values (0-1)
x_train, x_test = x_train / 255.0, x_test / 255.0

# Reshape to match the input shape for CNN (28x28x1)
x_train = x_train.reshape((-1, 28, 28, 1))
x_test = x_test.reshape((-1, 28, 28, 1))

# Print dataset shape
print(f"Train shape: {x_train.shape}, Test shape: {x_test.shape}")
```

- **Output**:

```
PS C:\Users\banke\Desktop\ITAI 3377\Module 03\New folder> & C:/Users/banke/anaconda3/python.exe "c:/Users/banke/Desktop/ITAI 3377/Module 03/New folder/deploy_mnist.py"
2025-02-14 20:33:49.385876: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-02-14 20:33:55.136925: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
Train shape: (60000, 28, 28, 1), Test shape: (10000, 28, 28, 1)
```

Step 3 – Train a Simple AI Model:

- **Define and Compile the Model**:
- ***Input***:

```python
# Define a simple CNN model
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

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

# Print model summary
model.summary()
```

- ***Output*:**

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
flatten (Flatten)	(None, 5408)	0
dense (Dense)	(None, 128)	692,352
dense_1 (Dense)	(None, 10)	1,290

Total params: 693,962 (2.65 MB)
Trainable params: 693,962 (2.65 MB)
Non-trainable params: 0 (0.00 B)

- **Train the Model:**

- ***Input*:**

```python
# Train the model
model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))

# Save the trained model
model.save("my_model.keras")
print("Model training complete and saved as my_model.keras")
```

- ***Output*:**

```
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 12s 6ms/step - accuracy: 0.9130 - loss: 0.2909 - val_accuracy: 0.9796
 - val_loss: 0.0601
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 6ms/step - accuracy: 0.9833 - loss: 0.0545 - val_accuracy: 0.9835
 - val_loss: 0.0459
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.9912 - loss: 0.0311 - val_accuracy: 0.9867
 - val_loss: 0.0405
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.9943 - loss: 0.0181 - val_accuracy: 0.9869
 - val_loss: 0.0411
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.9955 - loss: 0.0135 - val_accuracy: 0.9862
 - val_loss: 0.0466
Model training complete and saved as my_model.keras
```

Step 4 - Convert and Deploy the Model:

- **Convert the Model to TFLite:**

- ***Input*:**

```
# Convert the trained model to TensorFlow Lite format
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the TFLite model
with open("model.tflite", "wb") as f:
    f.write(tflite_model)

print("Model converted to TensorFlow Lite format and saved as model.tflite")
```

- ***Output*:**

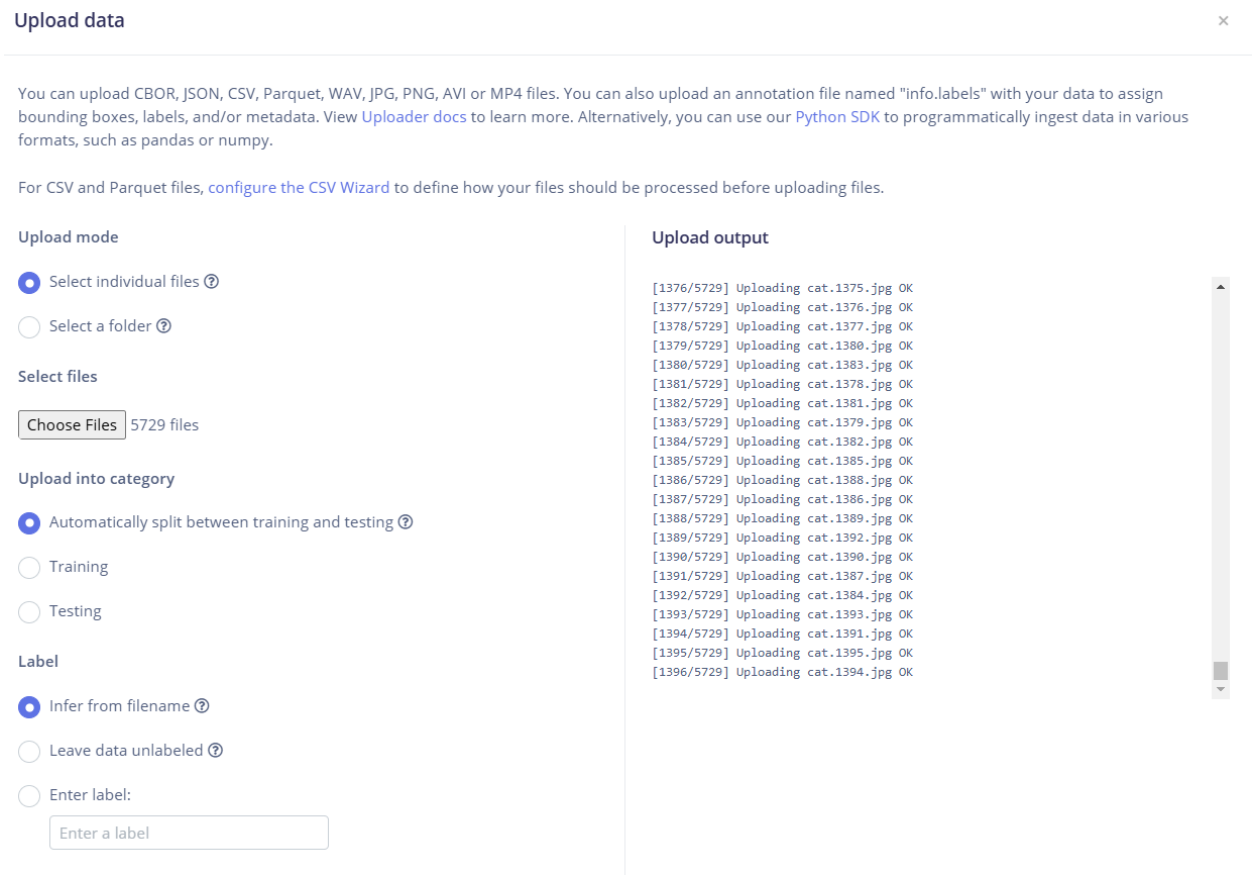
```
Saved artifact at 'C:\Users\banke\AppData\Local\Temp\tmpgykelhs1'. The following endpoints are available:

* Endpoint 'serve'
  args_0 (POSITIONAL_ONLY): TensorSpec(shape=(None, 28, 28, 1), dtype=tf.float32, name='keras_tensor')
Output Type:
  TensorSpec(shape=(None, 10), dtype=tf.float32, name=None)
Captures:
  2361092020688: TensorSpec(shape=(), dtype=tf.resource, name=None)
  2361092021648: TensorSpec(shape=(), dtype=tf.resource, name=None)
  2361092022992: TensorSpec(shape=(), dtype=tf.resource, name=None)
  2361092021840: TensorSpec(shape=(), dtype=tf.resource, name=None)
  2361092022032: TensorSpec(shape=(), dtype=tf.resource, name=None)
  2361092023568: TensorSpec(shape=(), dtype=tf.resource, name=None)
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
W0000 00:00:1739587894.768403   45244 tf_tfl_flatbuffer_helpers.cc:365] Ignored output_format.
W0000 00:00:1739587894.769467   45244 tf_tfl_flatbuffer_helpers.cc:368] Ignored drop_control_dependency.
2025-02-14 20:51:34.773673: I tensorflow/cc/saved_model/reader.cc:83] Reading SavedModel from: C:\Users\banke\AppData\Lo
cal\Temp\tmpgykelhs1
2025-02-14 20:51:34.777856: I tensorflow/cc/saved_model/reader.cc:52] Reading meta graph with tags { serve }
2025-02-14 20:51:34.778064: I tensorflow/cc/saved_model/reader.cc:147] Reading SavedModel debug info (if present) from:
C:\Users\banke\AppData\Local\Temp\tmpgykelhs1
I0000 00:00:1739587894.786496   45244 mlir_graph_optimization_pass.cc:401] MLIR V1 optimization pass is not enabled
2025-02-14 20:51:34.788322: I tensorflow/cc/saved_model/loader.cc:236] Restoring SavedModel bundle.
2025-02-14 20:51:34.856389: I tensorflow/cc/saved_model/loader.cc:220] Running initialization op on SavedModel bundle at
 path: C:\Users\banke\AppData\Local\Temp\tmpgykelhs1
2025-02-14 20:51:34.870901: I tensorflow/cc/saved_model/loader.cc:466] SavedModel load for tags { serve }; Status: succe
ss: OK. Took 97819 microseconds.
2025-02-14 20:51:34.958644: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR crash repr
oducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
Model converted to TensorFlow Lite format and saved as model.tflite
```

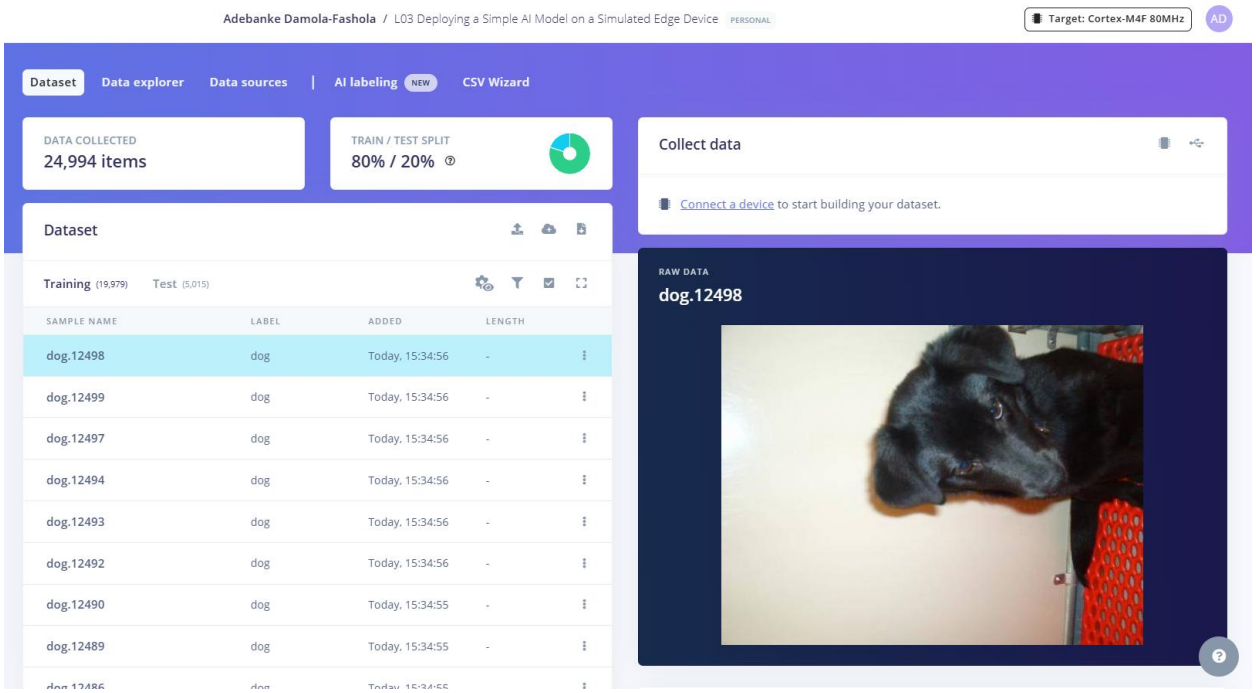
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   powershell

```
PS C:\Users\banke\Desktop\ITAI 3377\Module 03\New folder> npm install -g edge-impulse-cli
npm warn deprecated osenv@0.1.5: This package is no longer supported.
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a
good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated @zeit/dockerignore@0.0.5: "@zeit/dockerignore" is no longer maintained
npm warn deprecated move-concurrently@1.0.1: This package is no longer supported.
npm warn deprecated rimraf@2.7.1: Rimraf versions prior to v4 are no longer supported
npm warn deprecated npmlog@4.1.2: This package is no longer supported.
npm warn deprecated request-promise@4.2.4: request-promise has been deprecated because it extends the now deprecated request package,
 see https://github.com/request/request/issues/3142
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated are-we-there-yet@1.1.7: This package is no longer supported.
npm warn deprecated debug@4.1.1: Debug versions >=3.2.0 <3.2.7 || >=4 <4.3.1 have a low-severity ReDos regression when used in a Node
.js environment. It is recommended you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/debug/issues/797)
npm warn deprecated debug@4.1.1: Debug versions >=3.2.0 <3.2.7 || >=4 <4.3.1 have a low-severity ReDos regression when used in a Node
.js environment. It is recommended you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/debug/issues/797)
npm warn deprecated debug@4.1.1: Debug versions >=3.2.0 <3.2.7 || >=4 <4.3.1 have a low-severity ReDos regression when used in a Node
.js environment. It is recommended you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/debug/issues/797)
npm warn deprecated fs-write-stream-atomic@1.0.10: This package is no longer supported.
npm warn deprecated @xmldom/xmldom@0.8.8: this version has critical issues, please update to the latest version
npm warn deprecated gauge@2.7.4: This package is no longer supported.
npm warn deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain circumstance
s, which is known to be problematic.  See https://v8.dev/blog/math-random for details.
npm warn deprecated request@2.88.0: request has been deprecated, see https://github.com/request/request/issues/3142

changed 558 packages in 34s

49 packages are looking for funding
  run `npm fund` for details
```

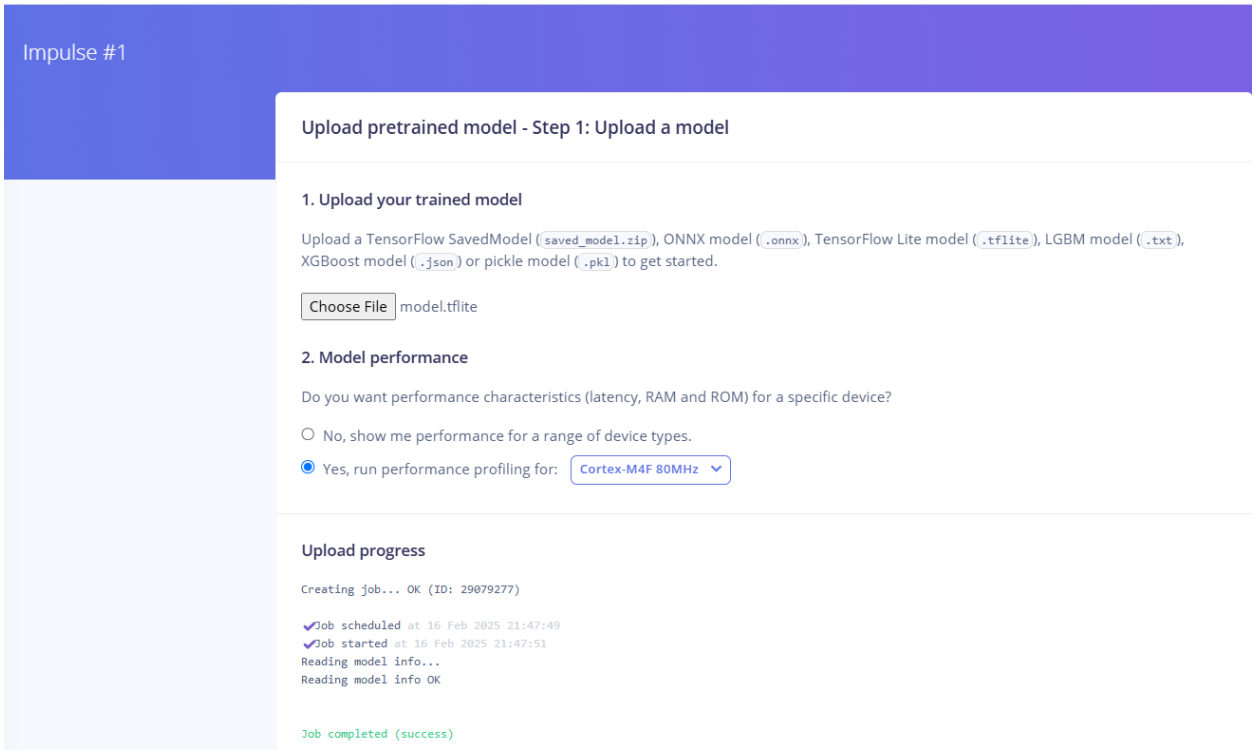
```
PS C:\Users\banke\Desktop\ITAI 3377\Module 03\New folder> edge-impulse-uploader --api-key ei_ce852fd77e5bf0a36eb2c438fa97f5135af6fdef
e4e2cf81bfebd2ee92649a49 model.tflite
Edge Impulse uploader v1.30.4
Endpoints:
    API:         https://studio.edgeimpulse.com
    Ingestion:   https://ingestion.edgeimpulse.com

Upload configuration:
    Label:       Not set, will be inferred from file name
    Category:    training
Cannot handle this file, only .wav, .cbor, .json, .jpg, .jpeg, .png, .csv, .txt, .mp4, .avi supported: model.tflite
```

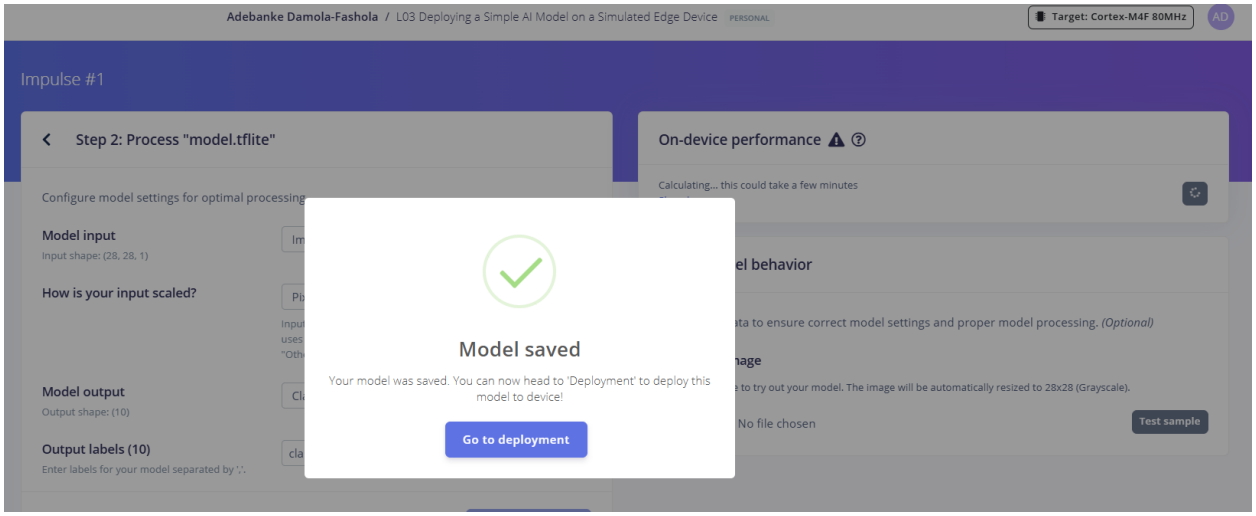
- **Upload the dataset to Edge Impulse:**

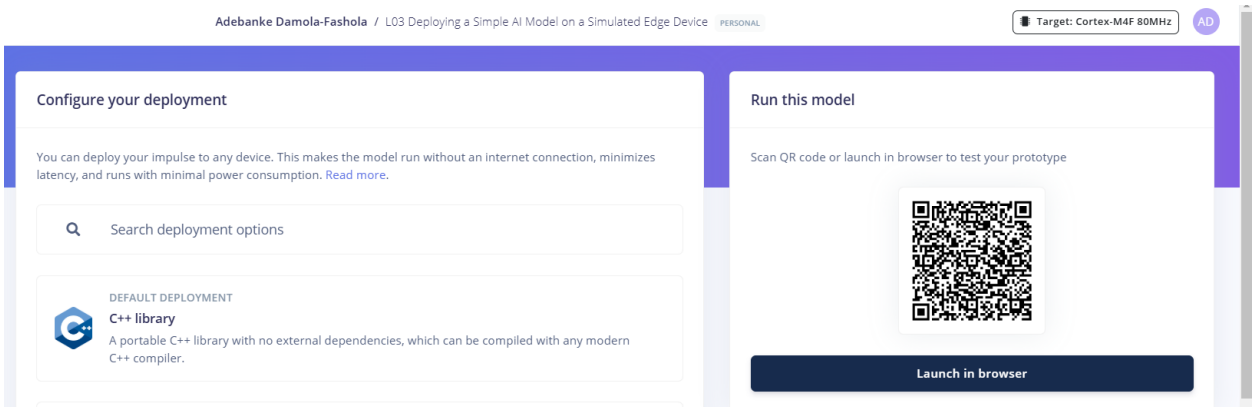


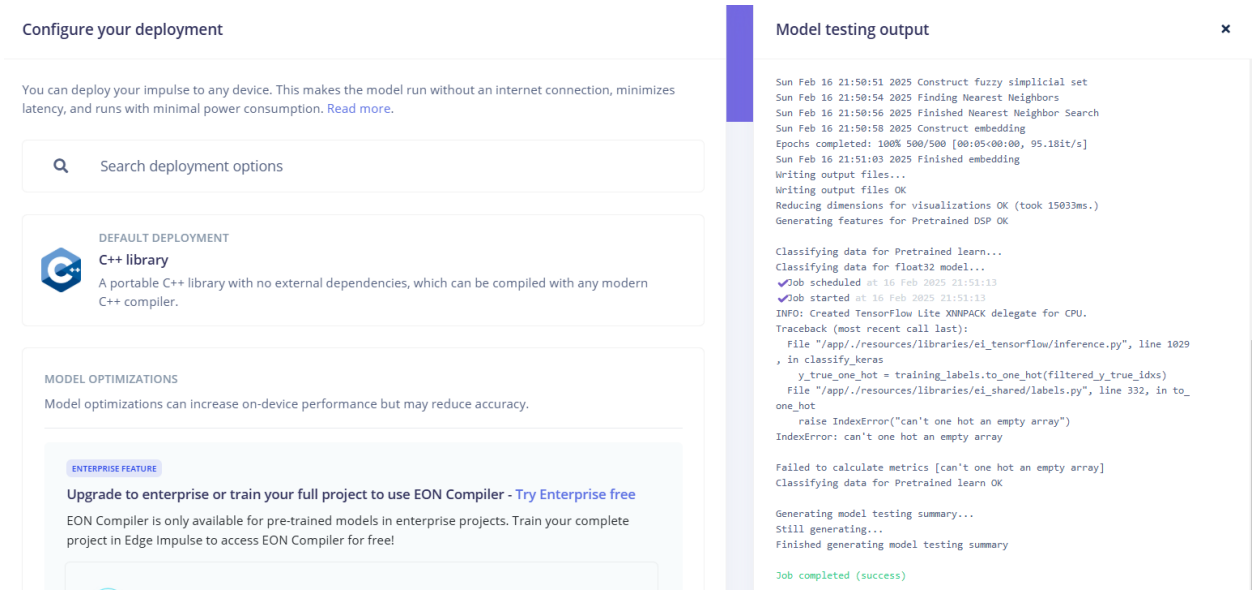


- **Upload the model to Edge Impulse:**

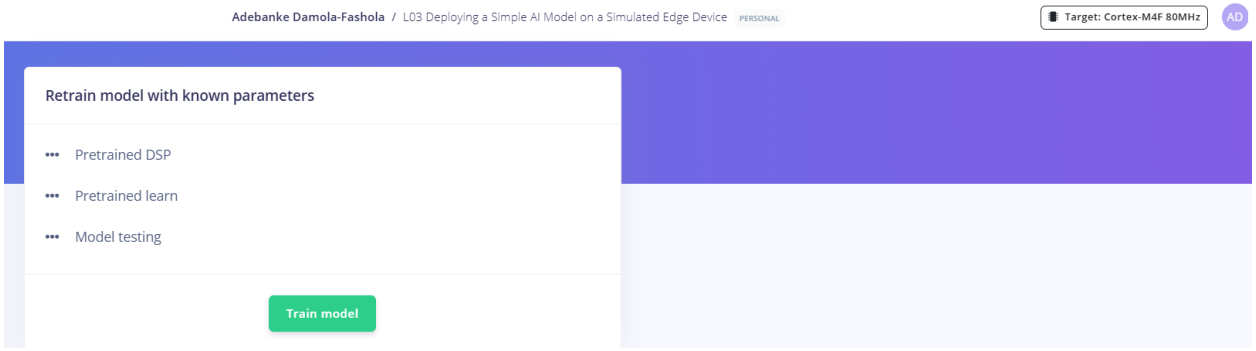


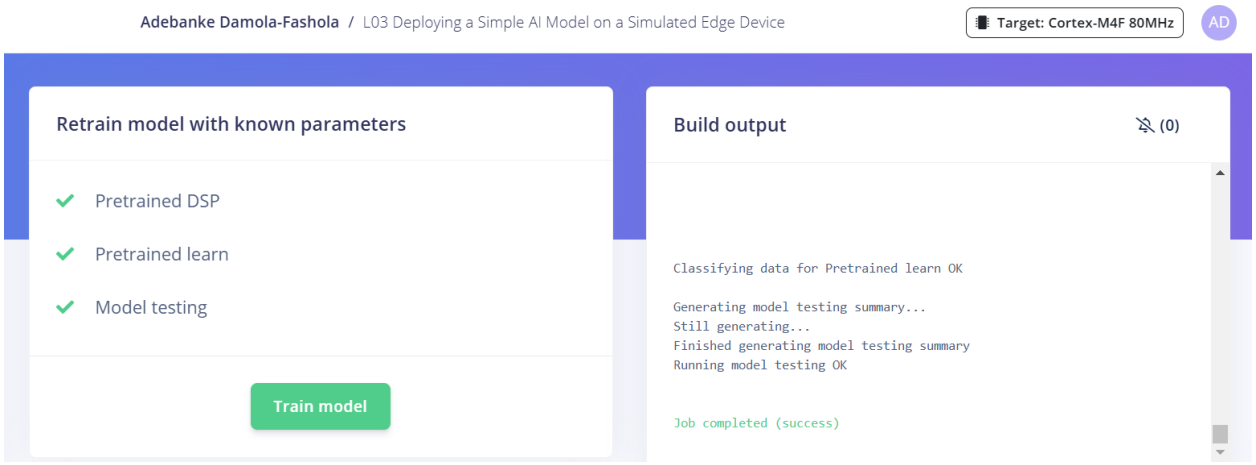

Adebanke Damola-Fashola / L03 Deploying a Simple AI Model on a Simulated Edge Device
PERSONAL
Target: Cortex-M4F 80MHz
AD
Impulse #1
Step 2: Process "model.tflite"
Configure model settings for optimal processing
Model input
Input shape: (28, 28, 1)
How is your input scaled?
Model output
Output shape: (10)
Output labels (10)
Enter labels for your model separated by ','.
On-device performance
Calculating... this could take a few minutes
el behavior
ata to ensure correct model settings and proper model processing. (Optional)
nage
to try out your model. The image will be automatically resized to 28x28 (Grayscale).
No file chosen
Test sample
Model saved
Your model was saved. You can now head to 'Deployment' to deploy this model to device!
Go to deployment


Adebanke Damola-Fashola / L03 Deploying a Simple AI Model on a Simulated Edge Device
PERSONAL
Target: Cortex-M4F 80MHz
AD
Configure your deployment
You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. Read more.
Search deployment options
DEFAULT DEPLOYMENT
C++ library
A portable C++ library with no external dependencies, which can be compiled with any modern C++ compiler.
Run this model
Scan QR code or launch in browser to test your prototype
Launch in browser

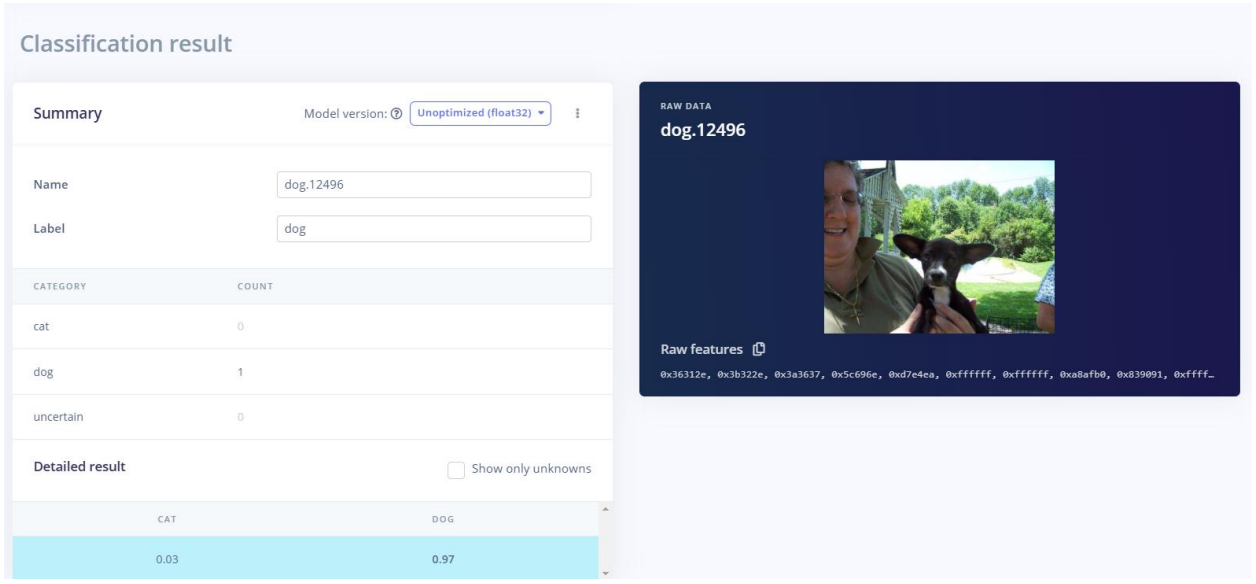

Configure your deployment
You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. Read more.
Search deployment options
DEFAULT DEPLOYMENT
C++ library
A portable C++ library with no external dependencies, which can be compiled with any modern C++ compiler.
MODEL OPTIMIZATIONS
Model optimizations can increase on-device performance but may reduce accuracy.
ENTERPRISE FEATURE
Upgrade to enterprise or train your full project to use EON Compiler - Try Enterprise free
EON Compiler is only available for pre-trained models in enterprise projects. Train your complete project in Edge Impulse to access EON Compiler for free!
Model testing output
Sun Feb 16 21:50:51 2025 Construct fuzzy simplicial set
Sun Feb 16 21:50:54 2025 Finding Nearest Neighbors
Sun Feb 16 21:50:56 2025 Finished Nearest Neighbor Search
Sun Feb 16 21:50:58 2025 Construct embedding
Epochs completed: 100% 500/500 [00:05<00:00, 95.18it/s]
Sun Feb 16 21:51:03 2025 Finished embedding
Writing output files...
Writing output files OK
Reducing dimensions for visualizations OK (took 15033ms.)
Generating features for Pretrained DSP OK
Classifying data for Pretrained learn...
Classifying data for float32 model...
Job scheduled at 16 Feb 2025 21:51:13
Job started at 16 Feb 2025 21:51:13
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Traceback (most recent call last):
File "/app/./resources/libraries/ei_tensorflow/inference.py", line 1029
, in classify_keras
y_true_one_hot = training_labels.to_one_hot(filtered_y_true_idxs)
File "/app/./resources/libraries/ei_shared/labels.py", line 332, in to_
one_hot
raise IndexError("can't one hot an empty array")
IndexError: can't one hot an empty array
Failed to calculate metrics [can't one hot an empty array]
Classifying data for Pretrained learn OK
Generating model testing summary...
Still generating...
Finished generating model testing summary
Job completed (success)

- **Retrain the model on Edge Impulse:**

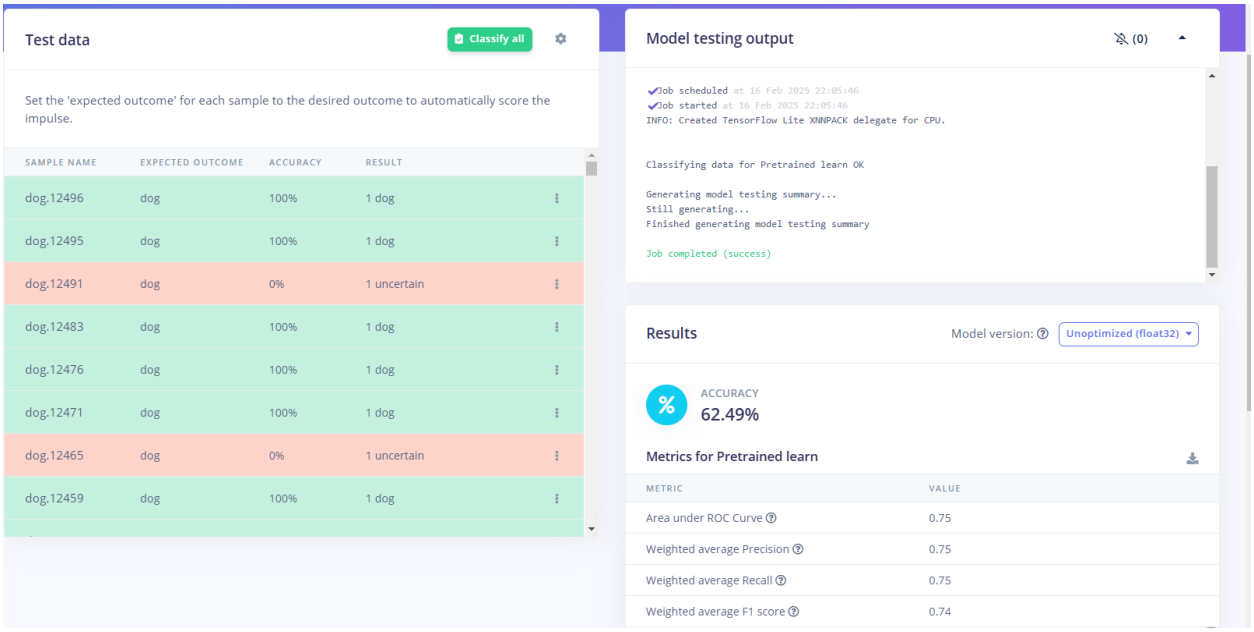




- **Use live classification to test:**



Step 5 - Test and Validate the Model:



Below is the breakdown of the key insights of a classification model for distinguishing between cats and dogs on Edge Impulse:

- **Model Performance Overview**:

Accuracy - 62.49%: The model correctly classified 62.49% of test samples.

ROC-AUC Score - 0.75: It suggests a moderately strong ability to distinguish between classes.

Weighted Precision: 0.75

Weighted Recall: 0.75

Weighted F1 Score: 0.74

Precision, Recall, and F1 Score: These metrics indicate a decent balance between precision and recall, though they suggest some misclassification issues.

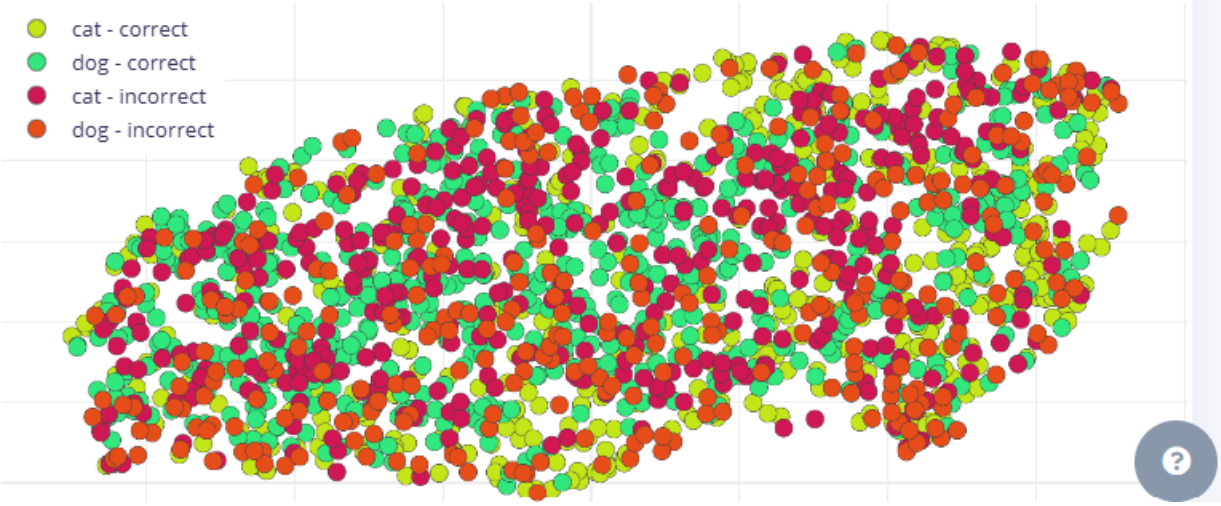
Metrics for Pretrained learn

METRIC	VALUE
Area under ROC Curve	0.75
Weighted average Precision	0.75
Weighted average Recall	0.75
Weighted average F1 score	0.74

Confusion matrix

	CAT	DOG	UNCERTAIN
CAT	56.9%	20.6%	22.5%
DOG	12.2%	68.2%	19.7%
F1 SCORE	0.67	0.72	

Feature explorer



- **Confusion Matrix Insights:**

Cat Classification: 56.9% correctly classified as cats, 20.6% misclassified as dogs, and 22.5% categorized as ‘uncertain.’

Dog Classification: 68.2% were correctly classified as dogs, 12.2% misclassified as cats, and 19.7% categorized as "uncertain."

F1 Scores: Cats: 0.67 Dogs: 0.72

The model performs slightly better on dogs than on cats.

- **Feature Explorer**:

The scatter plot visualizes feature space clustering. Green (correct classifications) and red (incorrect classifications) are mixed, suggesting overlapping features and contributing to classification challenges.

- **Key Takeaways and Areas for Improvement**:

The model's accuracy is moderate (62.49%). A high uncertainty rate (19-22%) suggests that the model struggles with certain cases. Dogs are classified better than cats based on F1 scores. Performance can be improved using a more balanced dataset with clean and accurately labeled data during training. Additionally, exploring hyperparameter tuning or adopting a more robust model architecture could help optimize classification accuracy and reduce misclassification rates.