Edge_Intelligence_Lab_MACSE604_

25MML0002

Adithyash BC

IN LAB 2

Part 1

To create a lightweight model(CNN/ANN) and train it on MNIST or other image datasets. Save the model using pickle and note the weight of the model

The model chosen was CNN and the dataset chosen was mnist for the following exercise

Code:

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.utils import to_categorical
import pickle
import numpy as np
```

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()

print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
```

```
X_train shape: (60000, 28, 28)
X_test shape: (10000, 28, 28)
```

```python
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0

# One-hot encode
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

print(f"X_train reshaped and normalized shape: {X_train.shape}")
print(f"y_train one-hot encoded shape: {y_train.shape}")
```

```
X_train reshaped and normalized shape: (60000, 28, 28, 1)
y_train one-hot encoded shape: (60000, 10)
```

```python
# Build the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
]:
```

```python
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_2 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten_1 (Flatten)	(None, 1600)	0
dense_2 (Dense)	(None, 128)	204,928
dense_3 (Dense)	(None, 10)	1,290

Total params: 225,034 (879.04 KB)

Trainable params: 225,034 (879.04 KB)

Non-trainable params: 0 (0.00 B)

```python
#training
history = model.fit(X_train, y_train, epochs=5, batch_size=32, validation_split=0.1)
# Evaluate
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Loss: {loss:.4f}")
print(f"Test Accuracy: {accuracy:.4f}")
```

```
Epoch 1/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 54s 31ms/step - accuracy: 0.9022 - loss: 0.3230 - val_accuracy: 0.9847 - val_loss: 0.0570
Epoch 2/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 51s 30ms/step - accuracy: 0.9852 - loss: 0.0499 - val_accuracy: 0.9858 - val_loss: 0.0502
Epoch 3/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 82s 30ms/step - accuracy: 0.9906 - loss: 0.0313 - val_accuracy: 0.9903 - val_loss: 0.0361
Epoch 4/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 51s 30ms/step - accuracy: 0.9929 - loss: 0.0226 - val_accuracy: 0.9900 - val_loss: 0.0366
Epoch 5/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 50s 29ms/step - accuracy: 0.9950 - loss: 0.0157 - val_accuracy: 0.9895 - val_loss: 0.0368
Test Loss: 0.0360
Test Accuracy: 0.9890
```

```python
model_filename = 'mnist_cnn.pkl'
with open(model_filename, 'wb') as file:
    pickle.dump(model, file)

print(f"Model saved to {model_filename} using pickle.")
```

```
Model saved to mnist_cnn.pkl using pickle.
```

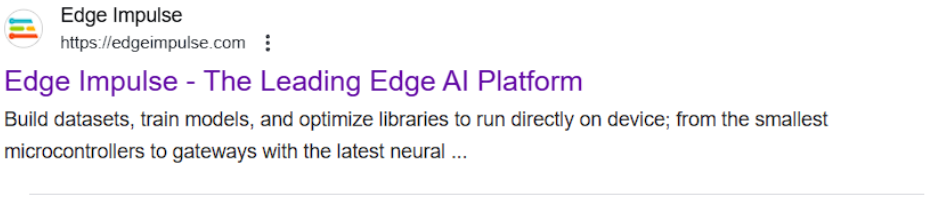
The weight/ size of the model can be found by navigating to the desired model via the Jupyter notebook interface

The size of the model is 2.74MB

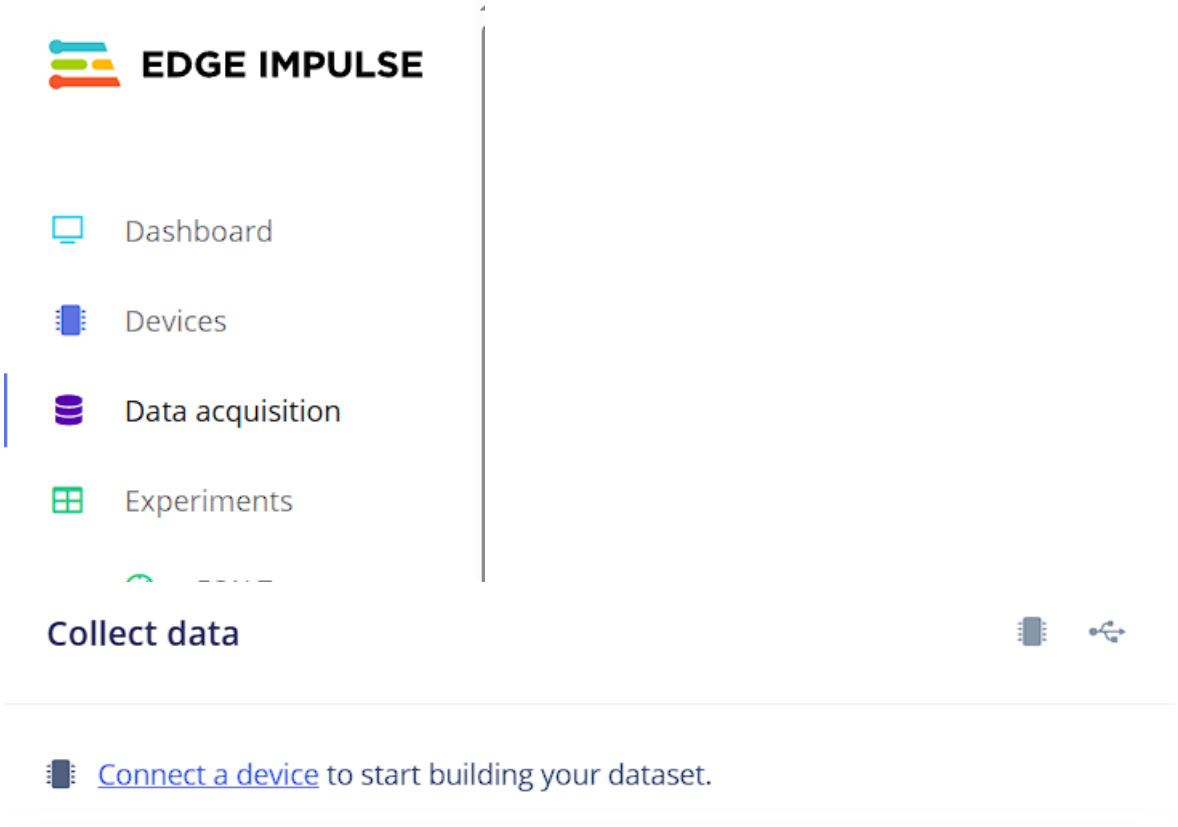
Part 2

Create an edge impulse account and create an image dataset by capturing images via your cellphone

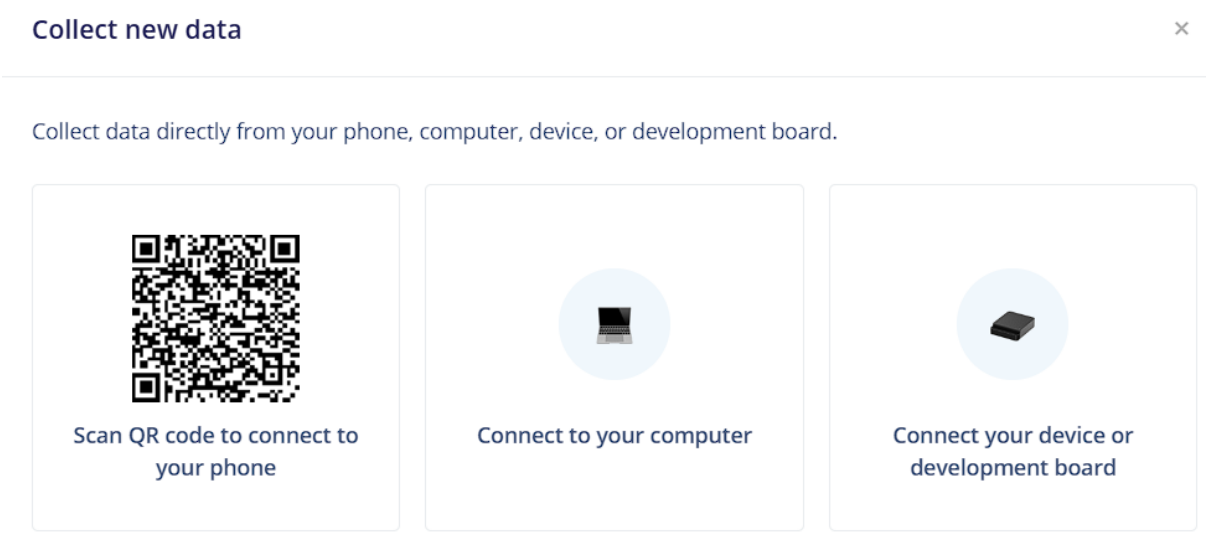
We must first navigate to edgeimpulse.com and sign up for an account



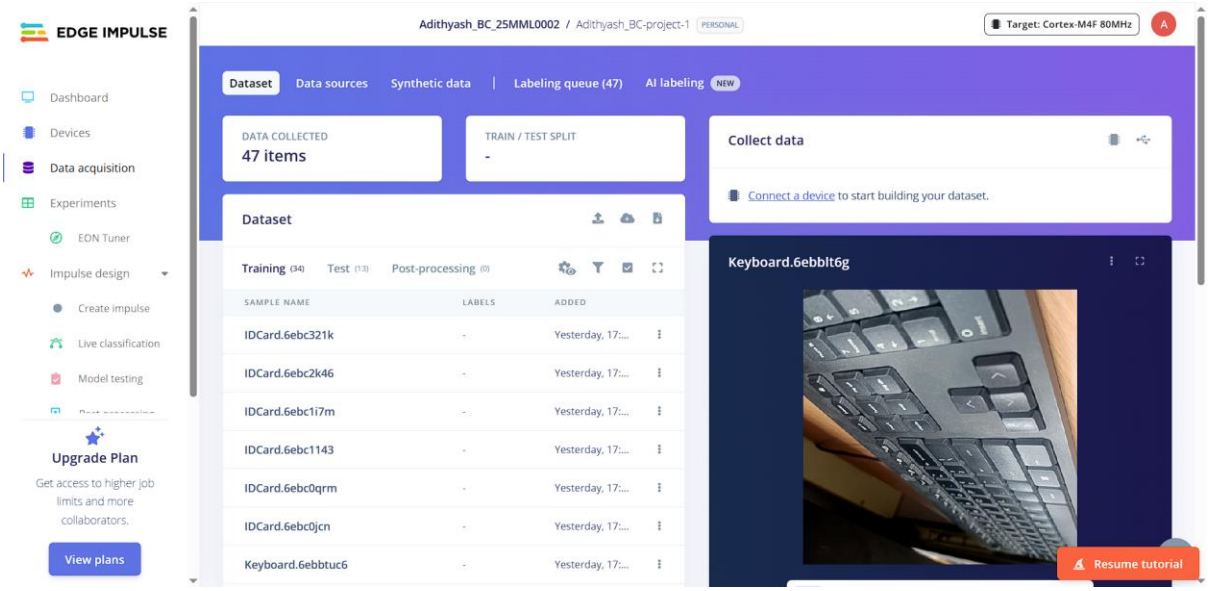
After filling in the required details we navigate to the data acquisition tab and click on “Connect a Device” which will prompt us with three options.



We pick the QR code option/ Mobile and scan the displayed QR code to connect our mobile device to the platform via an API



After that we may freely take photos of objects we want in our dataset and label them. The final view should look a little something like this

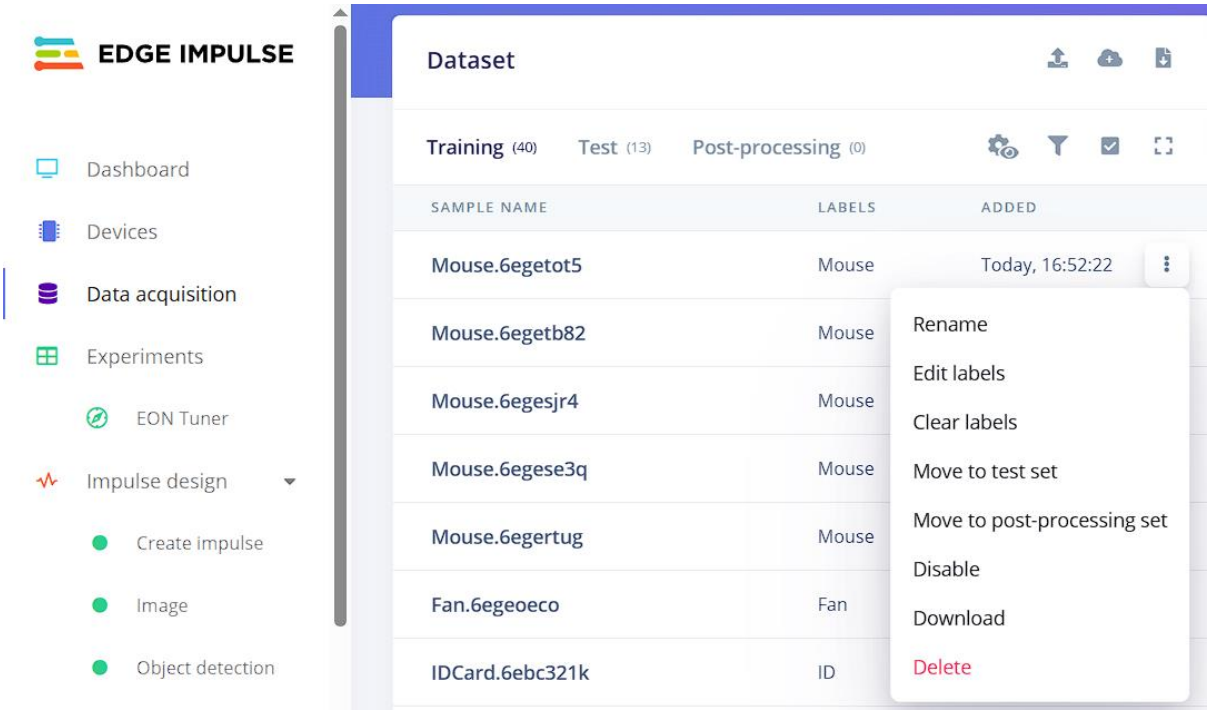


Edge In-Lab part 2

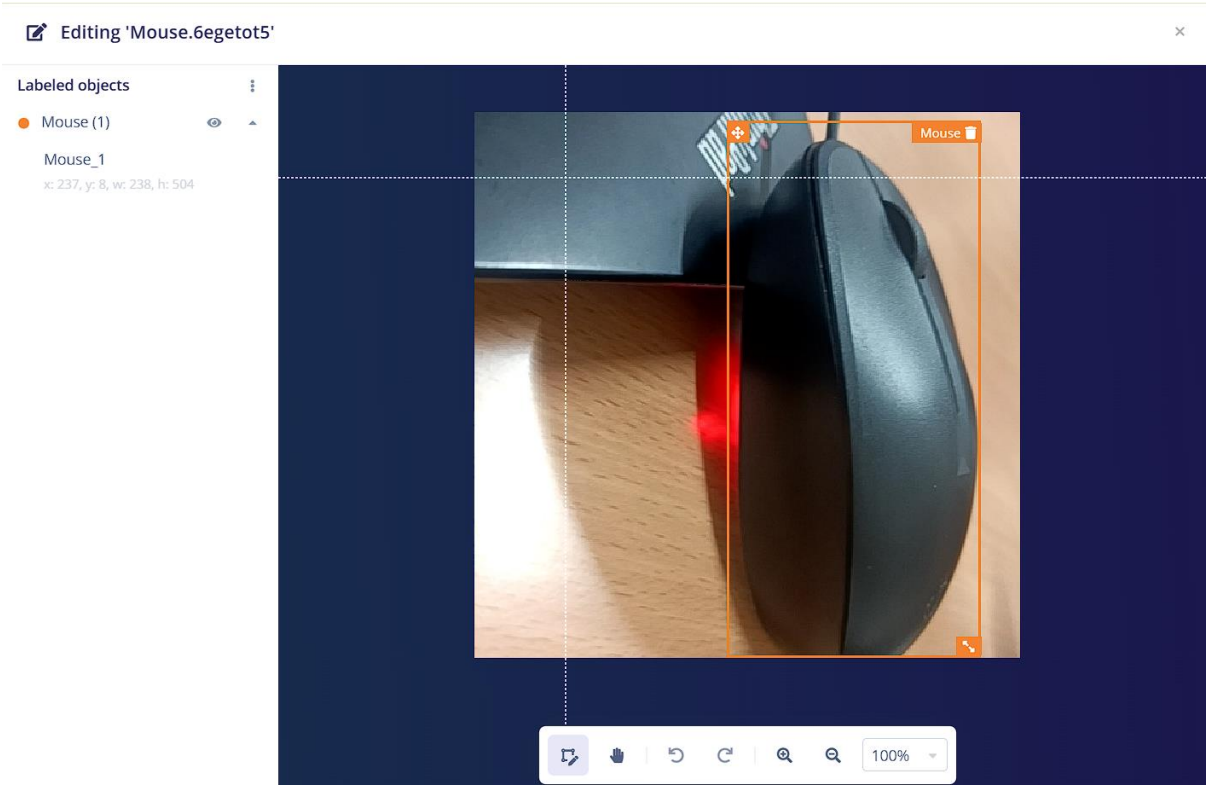
Adithyash BC

25MML0002

After the data has been acquired, it must be labelled. This can be done by navigating to the 3 dots near the images and selecting it and clicking on edit labels.



A window will pop up of the image. You can use the “plus” shaped crosser to box out the object to be labelled. Once boxed a pop up window will show up allowing you to change/make the label. This will help to better inform the model of what it is looking for.

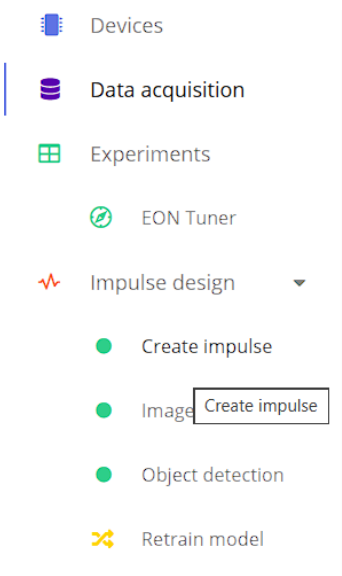


Repeat this process with all the images in the dataset

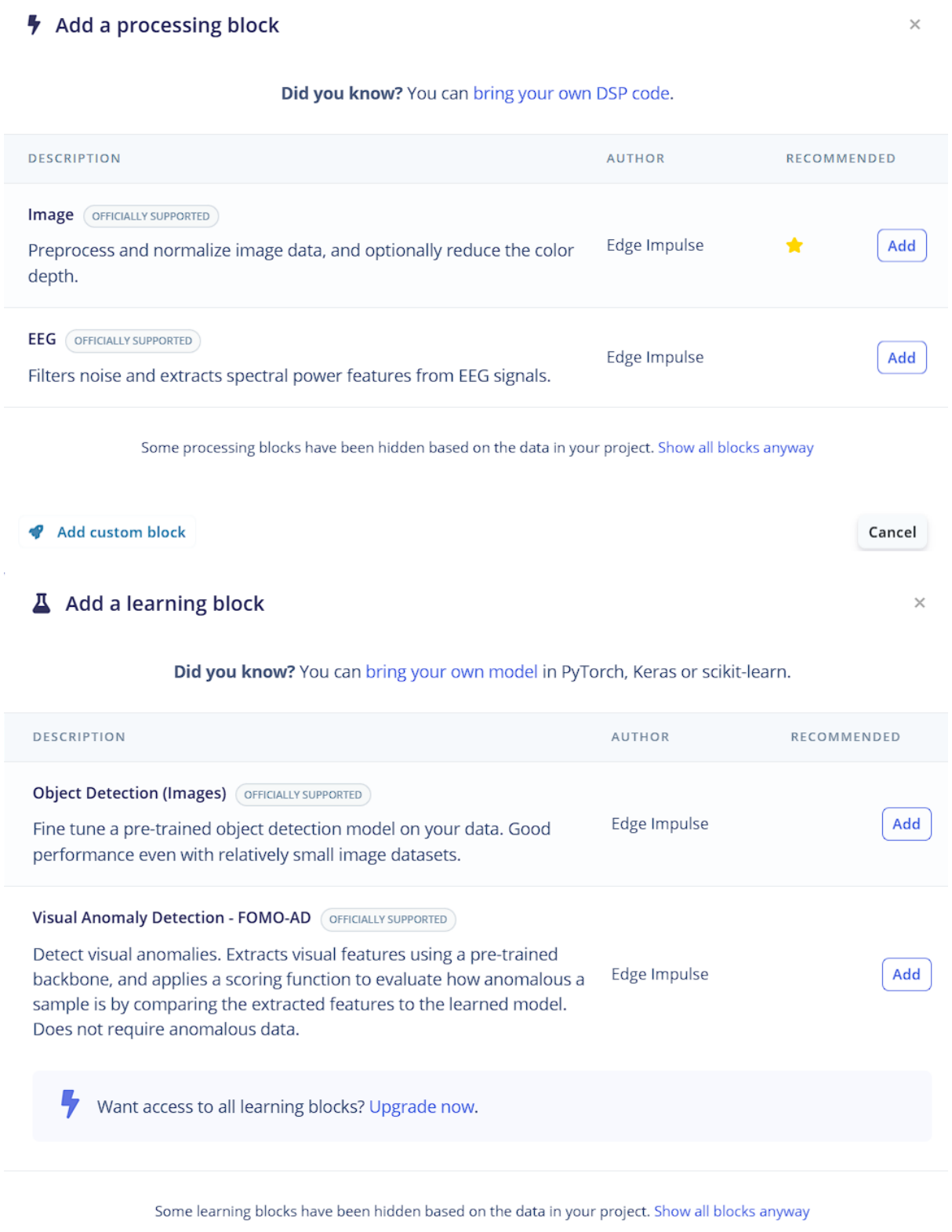
You will know if the label has been placed by looking at the “label” column in the dataset view

Ensure that the dataset is split into training and testing. The provided infographics on top of the dataset should be able to guide you into better managing this split

Once the data has been optimally split, navigate to the impulse tab and select on create an impulse

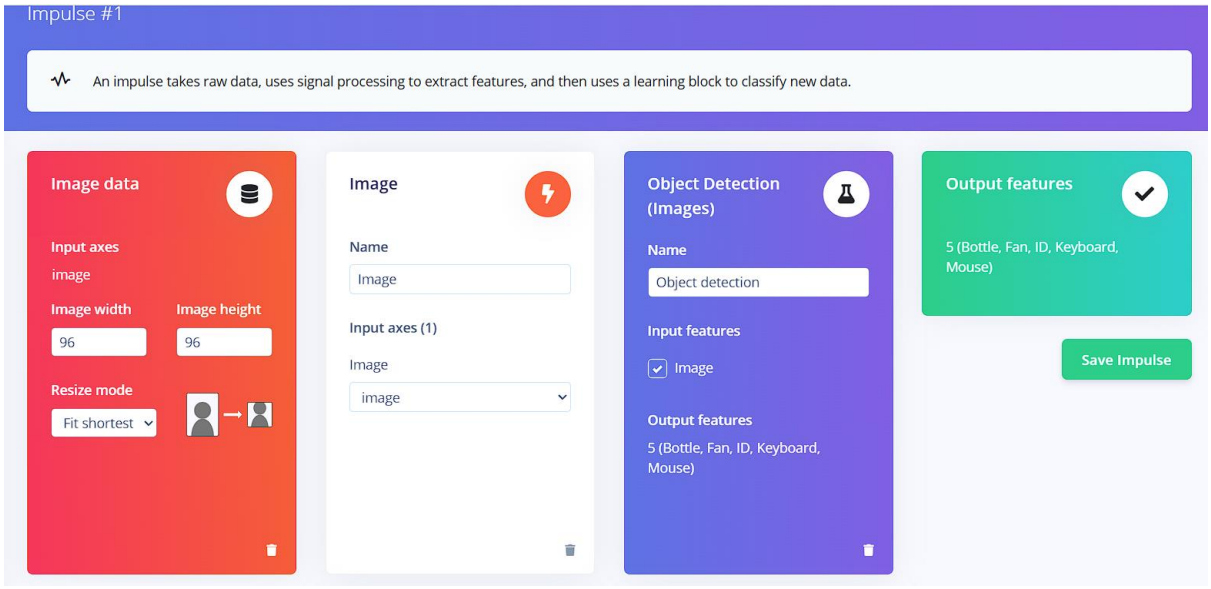


Click the “Add a processing block” button to select a format the model can process. Once that is done, click the “Add a learning block” button to select a model

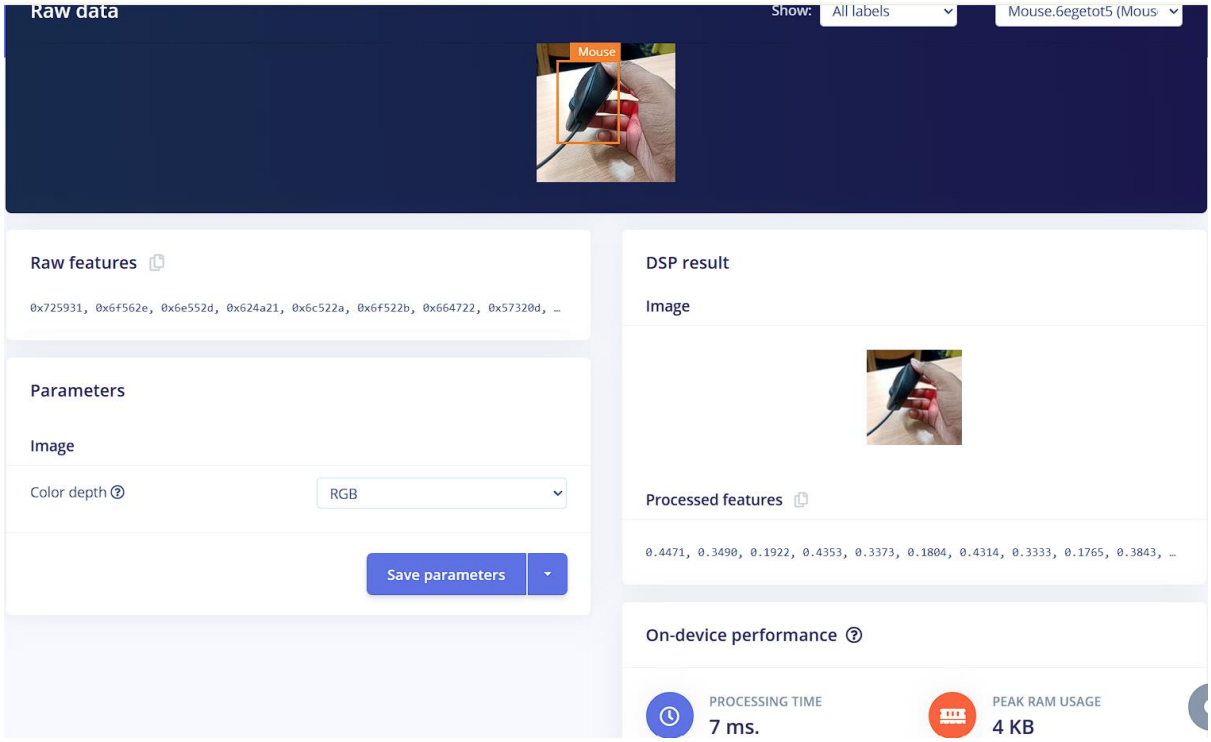


Once that is done, click on “the Save impulse” button to save the setup

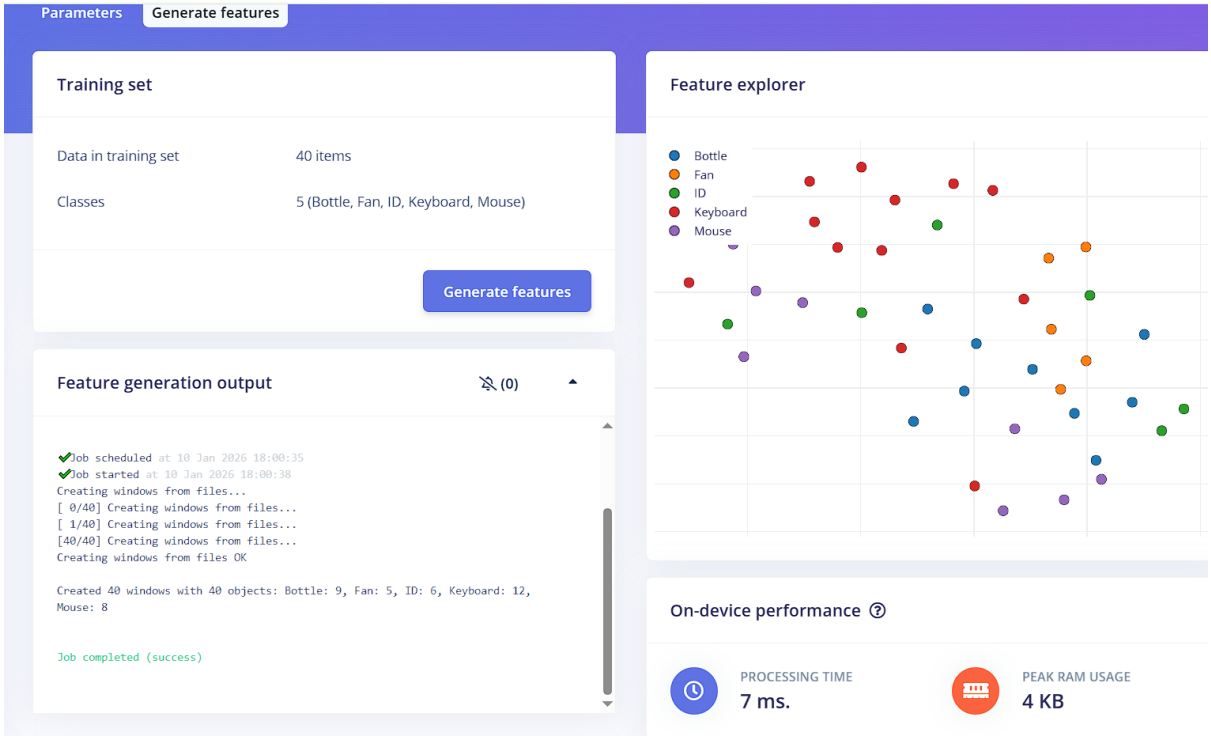
The final view should look like this



Navigate to the Image tab(below the impulse) and then save the parameters for the raw data. The parameters can be to process the image in RGB or in grayscale. Click on “save parameters” efore you proceed any further.

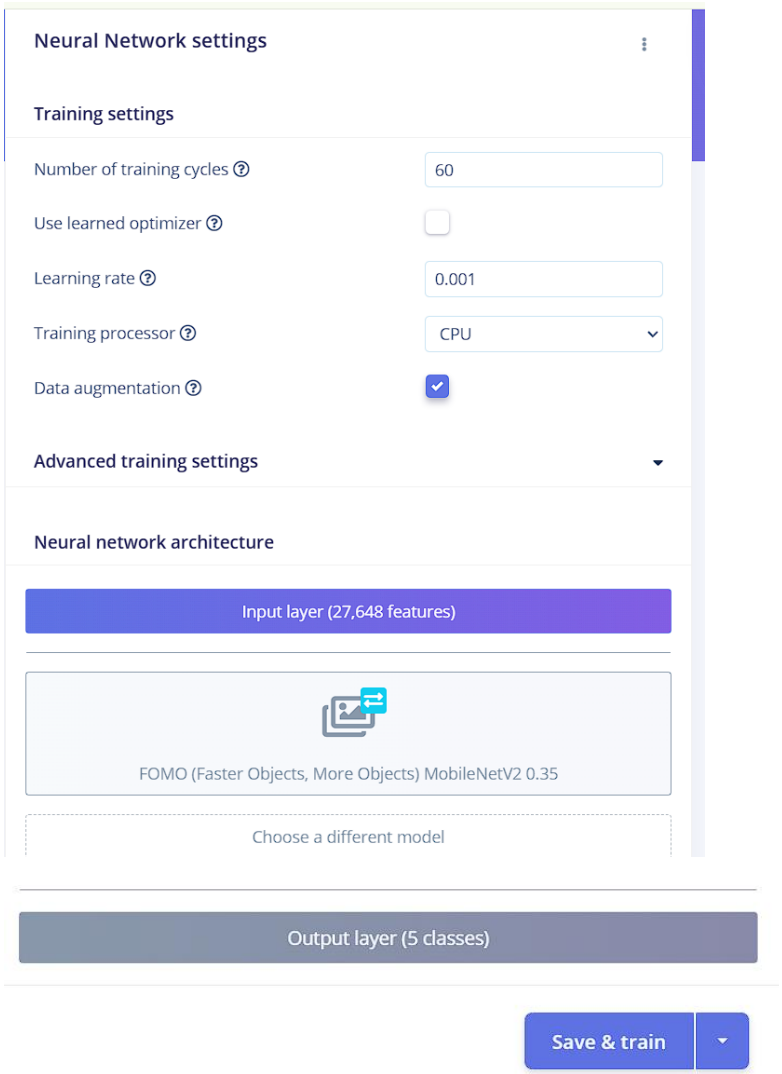


The tab will automatically navigate to the generate features tab. Once there you must click on generate features to load the features of your dataset. After it is done it should look a little something like this

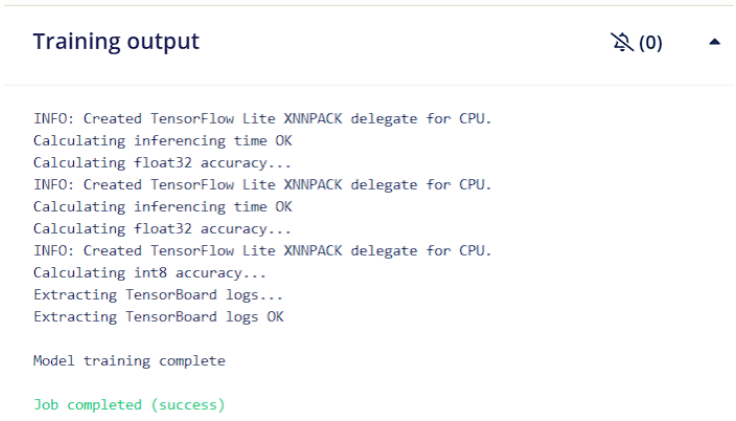


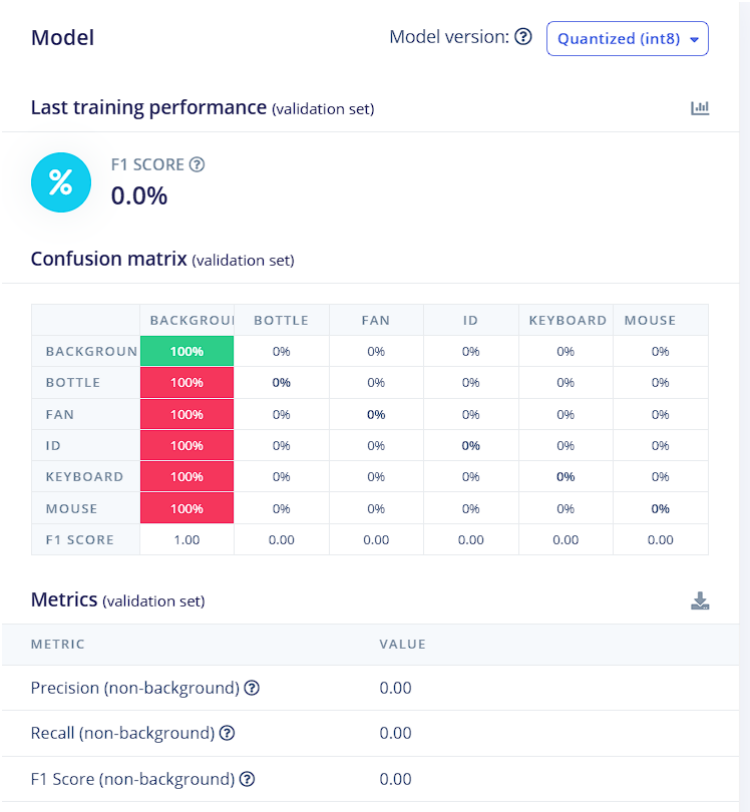
We can then navigate to the final tab, the “Object Detection” tab to run the model on our dataset.

There are various settings that can be found in the neural networks column that will allow you, the end user, to fine tune the model to your liking. Once all the nuances are dialled in(in this case we are just running it plain with no added changes), click on the “Save and train” button by navigating to the bottom of the page.



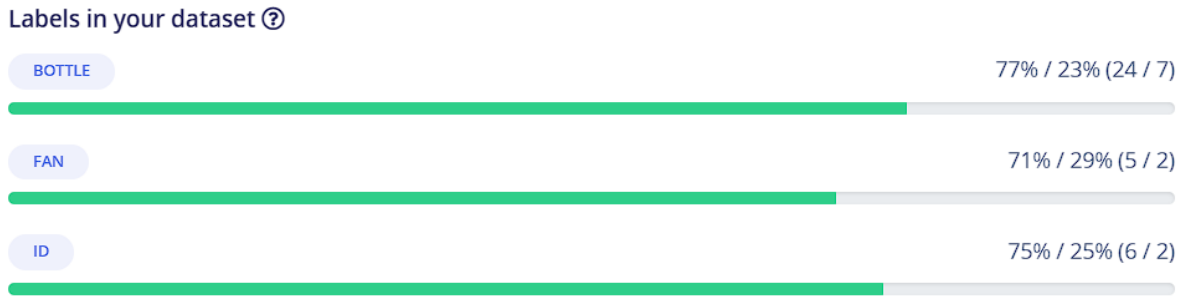
After a few minutes the Model should spit out the outputs, which can be found on the right-hand side of the page(under training outputs). Take note of the results and make corrections to the model or dataset if need be.





In our case the model gave us an F1 score of 0, which means that we will have to tune our dataset and model again to get our desired outputs.

The main reason why we were getting a low recall and f1 score was because our dataset was too small and thus the model did not have more to learn from. An easy fix to see if the model is working is to just increase the dataset size of one class. In this case I chose to increase the number of bottles



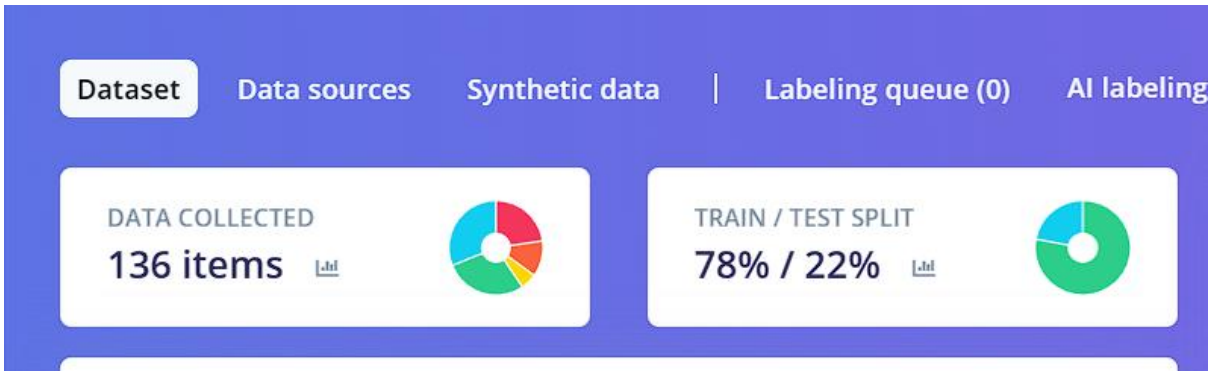
This small change gave us better results and showed me that we were on the right track as right after this we got this result

F1 SCORE
16.7%

Confusion matrix (validation set)

	BACKGROUI	BOTTLE	FAN	ID	KEYBOARD	MOUSE
BACKGROUN	100%	0%	0%	0%	0%	0%
BOTTLE	83.3%	16.7%	0%	0%	0%	0%
FAN	100%	0%	0%	0%	0%	0%
ID	100%	0%	0%	0%	0%	0%
KEYBOARD	100%	0%	0%	0%	0%	0%
MOUSE	100%	0%	0%	0%	0%	0%
F1 SCORE	1.00	0.29	0.00	0.00	0.00	0.00

This confirms about our dataset theory(for now). By adding more images to the dataset



And manually annotating it, we can increase the robustness of the model and the data thus giving us better results.

We can now cycle through the same previous steps of saving impulse and generating features and then training the model to get the newer results.

Last training performance (validation set)

F1 SCORE
45.2%

Confusion matrix (validation set)

	BACKGROUI	BOTTLE	FAN	ID	KEYBOARD	MOUSE
BACKGROUN	100%	0%	0%	0%	0%	0%
BOTTLE	100%	0%	0%	0%	0%	0%
FAN	100%	0%	0%	0%	0%	0%
ID	-	-	-	-	-	-
KEYBOARD	22.2%	0%	0%	0%	77.8%	0%
MOUSE	100%	0%	0%	0%	0%	0%
F1 SCORE	1.00	0.00	0.00		0.88	0.00

These results show us that the increased dataset pool is helping the model to better understand and recognize the images. The way to increase the F1 score is to use even more images and even better images.