Title page: EEDG/CE 6302 Lab Report 5
CE6302, Embedded Systems, 302 Laboratory - 83204
Lab Topic: - TinyML - Image Classification
Yuyang Hsieh, YXH230019
Lab partner name: Xiongtao Zhang
Group Number 4
Instructor name: Tooraj Nikoubin
TA name: Seyed Saeed
Date: 9/28/2024

1. Objective:

Purpose: The purpose of this lab is using TinyML and machine learning algorithms to build an image recognition system. By using the Espressif ESP-EYE (ESP32), the system will classify different objects, key and cookie, and when neither of them appear, the system will identify the image as unknown. This lab provides practical experience in data collection, signal processing, and deploying machine learning models on embedded devices for real-time image classification, and also makes use of existing repositories of dataset to build more powerful model.

Methods used: In part 1 of lab 5, by using Edge Impulse, an image classification TinyML model is trained, validated and deployed based on the data collected by taking photos using ESP-EYE (ESP32). In part 2 of the lab, by using Hugging Face, which contains millions of repositories regarding AL/ML, a pre-built data set is used to train a new image classification model.

Results: In part 1, after collecting the data, generating the model, and deploying the model, the device can successfully classify what object appears in the image. When put the key (or cookie) in front of the device's camera, it classified it as key (or cookie) with approximate 0.9 accuracy, and when put the keyboard or some object except key and cookie, the device classified it as unknown. In part 2, by using dataset from hugging face, and training on edge impulse, an image classification model to classify beans is built and tested with high accuracy.

Hardware used: Espressif ESP-EYE (ESP32)

Software used: Edge Impulse CLI, Hugging Face.

Major Conclusions: By collecting data, analyzing data, extracting features, training model, evaluating model and deploying model, the device can detect and distinguish the object between key, cookie and unknown(neither key nor cookie) in real time by taking images of it. And by using dataset from hugging face, a model to classify images of beans is built and tested in edge impulse.

2. Introduction:

2.1 Hardware and Software Background Information

Hardware Background:

Espressif ESP-EYE (ESP32): ESP-EYE is a development board for image recognition and audio processing, which can be used in various AIoT applications. It features an ESP32 chip, a 2-Megapixel camera and a microphone. ESP-EYE offers plenty of storage, with an 8 Mbyte PSRAM and a 4 Mbyte flash. It also supports image transmission via Wi-Fi and debugging through a Micro-USB port.

Software Background Information:

Edge Impulse: which is a cloud platform ushering in the future of embedded machine learning by empowering developers to create and optimize solutions with real-world data. It makes the process of building, deploying, and scaling embedded ML applications easier and faster than ever, unlocking massive value across every industry, with millions of developers making billions of devices smarter.

Hugging Face: The Hugging Face Hub is a platform with over 900k models, 200k datasets, and 300k demo apps (Spaces), all open source and publicly available, in an online platform where people can easily collaborate and build ML together. The Hub works as a central place where anyone can explore, experiment, collaborate, and build technology with Machine Learning.

2.2 Purpose of the Experiment

The purpose of lab 5 is using TinyML to perform image detection tasks on microcontrollers based on machine learning. The lab demonstrates a completed workflow of building image classification models. It includes collecting data from embedded device and upload to cloud platform, Edge Impulse; Processing signals and distinguishing between the three types of images (key, cookie, and unknown) with designing an impulse on cloud; Generating features based on previous data and training neural network model; After that, classifying the new Data with trained model to evaluate its performance; Finally, deploying the model back to device. Also, make use of existing dataset on image classification and train a new model from it, which creates a more powerful model that can deploy in more complex tasks with TinyML.

2.3 Summary of the Experiment

In lab 5, the flow of implementing image-classification TinyML model on microcontrollers is demonstrated. From data collection, signal processing to generating features, training model, verifying the model, and deployment, the procedures of applying TinyML are completed. After deployment, the device can successfully identify the object its camera captures and classify it into three types(key, cookie, or unknown). By applying TinyML, computation-intensive and memory-intensive applications are performed on the cloud, models and related applications are optimized to be able to run on the microcontrollers and embedded systems with low-power and real-time features. And by using Hugging Face, people can use any open-sourced dataset to train their model, which enables the ability to build more powerful TinyML models and perform more complex tasks on embedded systems.

2.4 Findings of the experiment

In the data collection part, the size(resolution) of the collected image is set to 64x64, which is relatively small. This makes it to be performed quickly while using less time, but it requires more accurate images. For example, when taking photos of a key, the background needs to be clear, and the key needs to be placed in the center of the image. Otherwise, the accuracy and performance of the model will be downgraded.

3. Explanation:

3.1 Experiment procedure

Part 1

To begin part 1, connect the Espressif ESP-EYE ESP32 development board to your Windows system. You will be installing the CP210x USB to UART Bridge drivers that enable your computer to talk to the ESP-EYE. Once you have installed the drivers and the board is detected, open a terminal and run edge-impulse-daemon. This will open a wizard asking you to log in to your Edge Impulse account. After logging in, select or create an Edge Impulse project. If you want to switch projects later, run this command again with the option --clean.

Next, build your dataset using the ESP-EYE camera. Set the camera sensor to capture images at 64x64 resolution. For the dataset, collect 50 images each of a banana, an apple, and 50 additional images of random objects not related to either fruit. Ensure that your dataset has a wide variety of images for balance. Upload this dataset to the Edge Impulse platform, and under Data Acquisition, verify that all images have been uploaded correctly.

If the dataset is prepared, an impulse can be designed by setting the image width and height to 96 x 96 pixels. The 'Images' and 'Transfer Learning (Images)' blocks will be added onto the impulse. The processing block can be configured after saving the impulse. It will select RGB color depth for the images in order to prepare the raw data for generating features. The features could be generated by resizing the dataset and applying the processing block to all images. This step will also provide a 3-D visualization of the dataset, which will enable one to view how the data collected is distributed.

Let the transfer learning model train a neural network now on this dataset. Under Transfer Learning, select a base model and put the number of training cycles as 100. Use a Learning Rate of 0.005, enable Data Augmentation and Validation Set as 20%., then click on Train Model to train your model. It will display accuracy results, a confusion matrix, and on-device performance metrics in its output when done.

We will also validate the model using the test dataset, since real-world cases depend on it. We separated the testing dataset from the training since overfitting could occur easily. Then, in the Model Testing section, select some samples for testing, classify them and run it. Once validated, an accuracy of about 75% could be seen depending on the dataset.

Finally, deploy the trained model to your ESP-EYE board. Under the Deployment menu, select 'Build firmware' and choose your device (Espressif ESP-EYE (ESP32)). After the firmware is built, open a terminal and run the command edge-impulse-run-impulse to execute the model on the device, allowing live image classification. Optionally, explore larger datasets, such as those from Hugging Face, to improve model performance and accuracy further.

Part 2

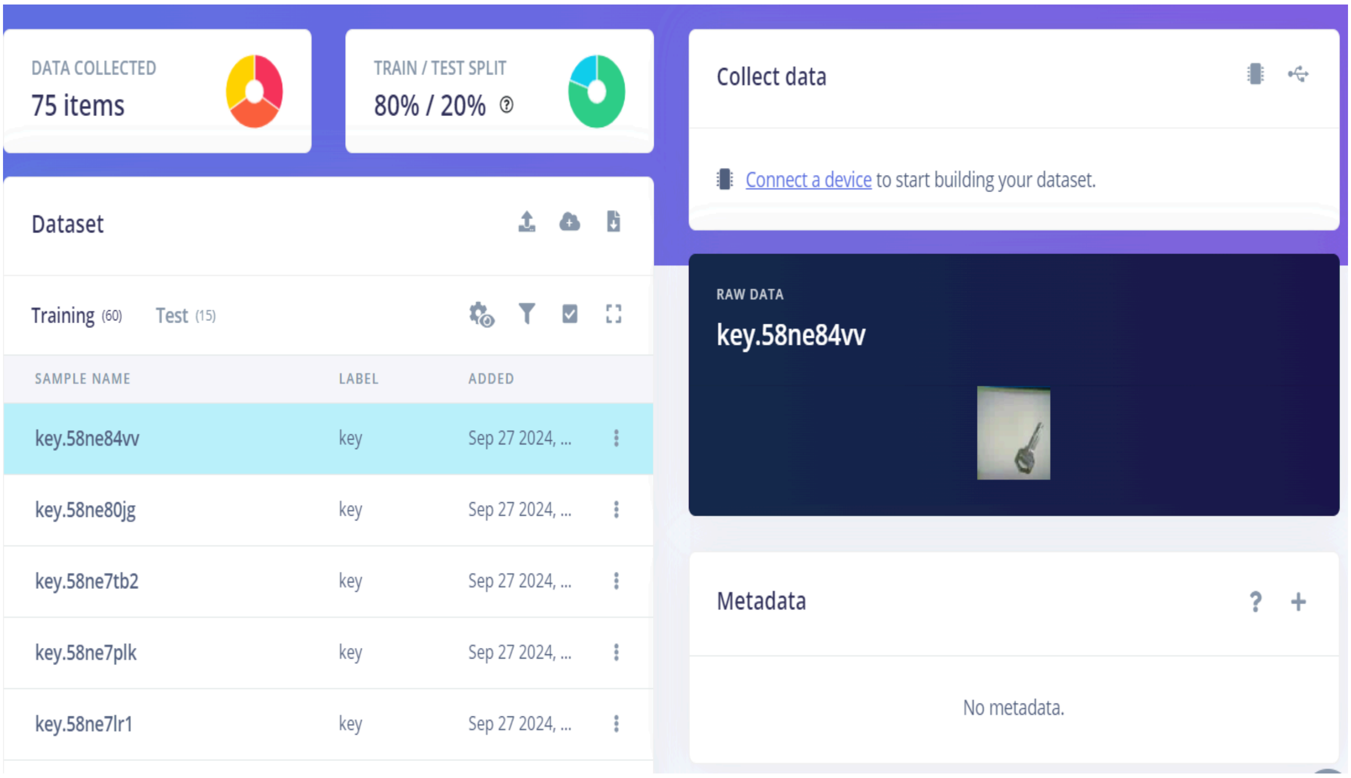
In part 2, the procedure focuses on image classification of plant diseases using the Hugging Face Beans dataset. First, you begin by installing Git Large File Storage (LFS) by executing the command `git lfs install`. Then, clone the beans dataset from Hugging Face by running `git clone https://huggingface.co/datasets/beans`. Once the dataset is downloaded, navigate into the cloned directory and unzip the file. This will reveal a folder structure with three subdirectories: Training, Testing, and Validation. These contain images corresponding to healthy beans, beans affected by angular leaf spot, and beans with bean rust, respectively.

Next, create a new project in Edge Impulse to perform the image classification. Within the Edge Impulse interface, select the Data Acquisition tab, where you will upload the dataset. Under the training section, upload the images categorized into Healthy, Angular Leaf Spot, and Bean Rust. Similarly, upload the testing images in the testing section.

Once the dataset is uploaded, you can create an impulse. Set the image parameters to 96x96 resolution, and utilize the MobileNetV2 algorithm with a width multiplier of 0.35. Configure the last layer of the model to have 16 neurons with a dropout rate of 0.1. For the training configuration, set the model to run for 120 epochs, with a validation set size of 10% and a learning rate of 0.0005.

After the model is trained, navigate to Model Testing and classify the test images using the trained model. This step will allow you to evaluate the model’s performance and accuracy in classifying the bean images along with their corresponding plant diseases.

3.2 Experiment Images for Part 1



DATA COLLECTED

75 items

TRAIN / TEST SPLIT

80% / 20%

Collect data

Connect a device to start building your dataset.

Dataset

Training (60) Test (15)

SAMPLE NAME	LABEL	ADDED
key.58ne84vv	key	Sep 27 2024, ...
key.58ne80jg	key	Sep 27 2024, ...
key.58ne7tb2	key	Sep 27 2024, ...
key.58ne7plk	key	Sep 27 2024, ...
key.58ne7lr1	key	Sep 27 2024, ...

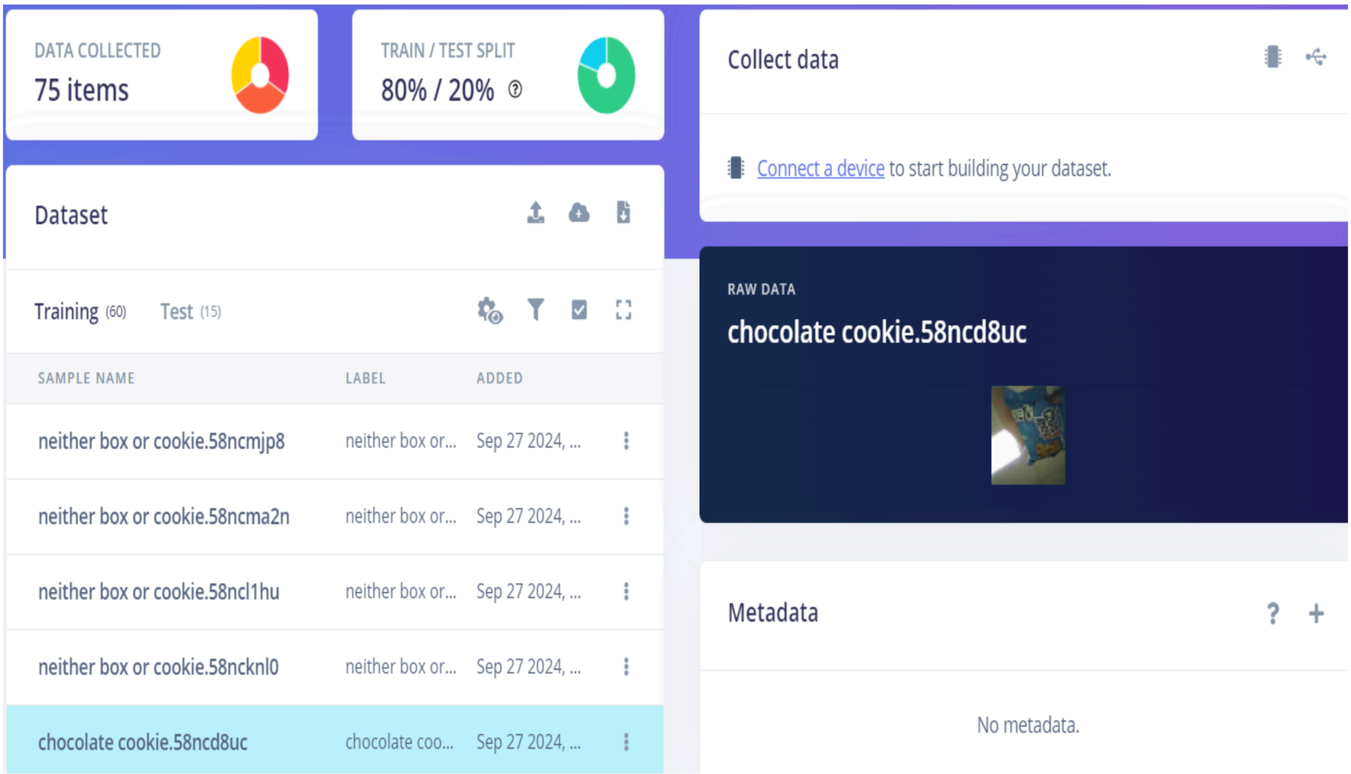
RAW DATA

key.58ne84vv

Metadata

No metadata.

Image 3.2-1: : images data for key



DATA COLLECTED
75 items

TRAIN / TEST SPLIT
80% / 20%

Collect data

Connect a device to start building your dataset.

Dataset

Training (60) Test (15)

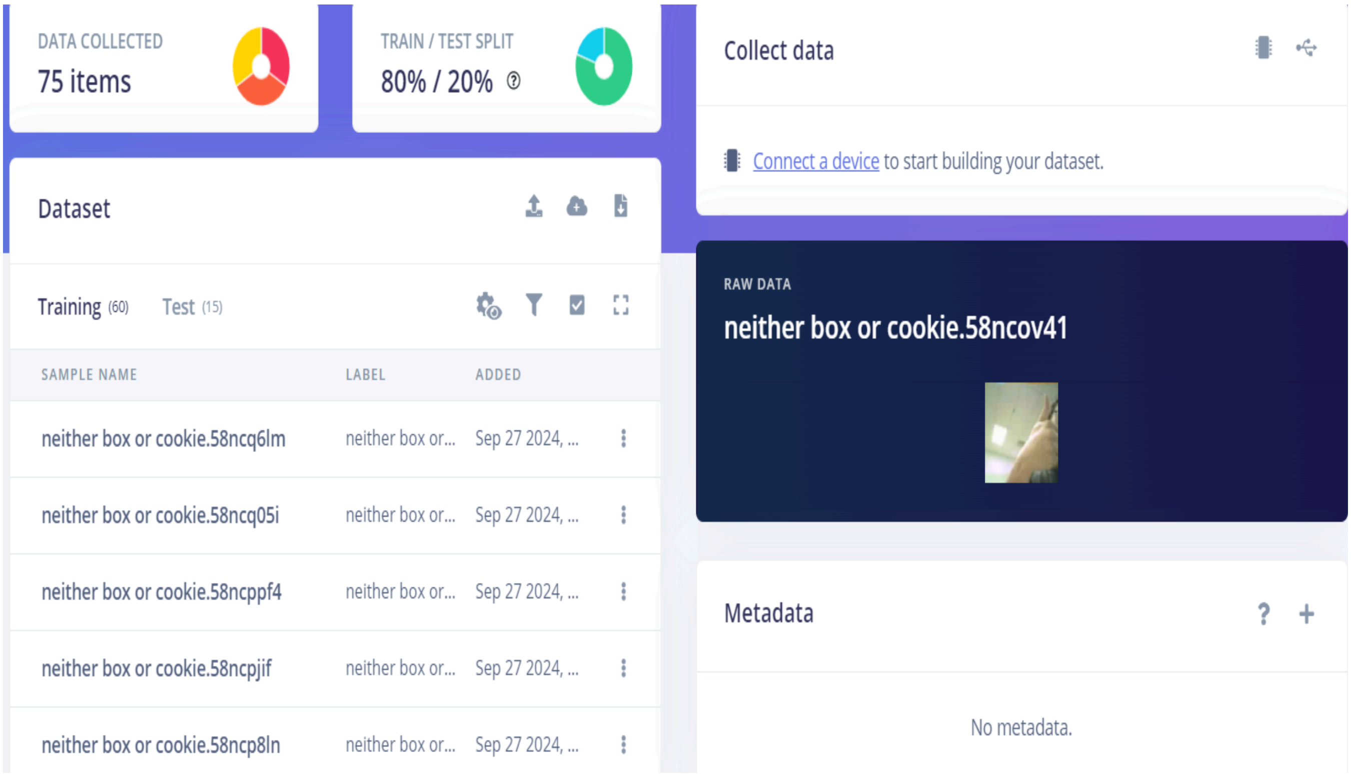
SAMPLE NAME	LABEL	ADDED
neither box or cookie.58ncmjp8	neither box or...	Sep 27 2024, ...
neither box or cookie.58ncma2n	neither box or...	Sep 27 2024, ...
neither box or cookie.58ncl1hu	neither box or...	Sep 27 2024, ...
neither box or cookie.58ncknl0	neither box or...	Sep 27 2024, ...
chocolate cookie.58ncd8uc	chocolate coo...	Sep 27 2024, ...

RAW DATA
chocolate cookie.58ncd8uc

Metadata

No metadata.

Image 3.2-2: : images data for cookie



DATA COLLECTED
75 items

TRAIN / TEST SPLIT
80% / 20%

Collect data

Connect a device to start building your dataset.

Dataset

Training (60) Test (15)

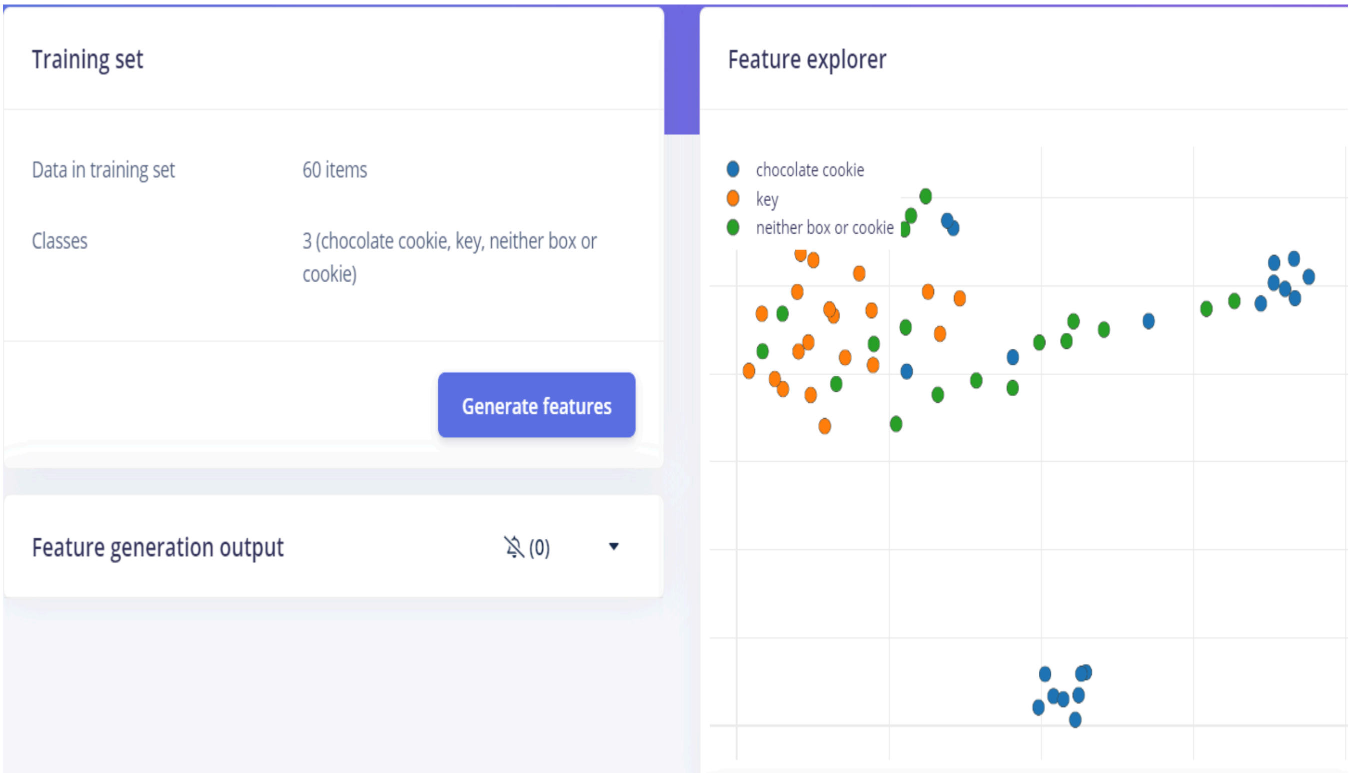
SAMPLE NAME	LABEL	ADDED
neither box or cookie.58ncq6lm	neither box or...	Sep 27 2024, ...
neither box or cookie.58ncq05i	neither box or...	Sep 27 2024, ...
neither box or cookie.58ncppf4	neither box or...	Sep 27 2024, ...
neither box or cookie.58ncpjif	neither box or...	Sep 27 2024, ...
neither box or cookie.58ncp8ln	neither box or...	Sep 27 2024, ...

RAW DATA
neither box or cookie.58ncov41

Metadata

No metadata.

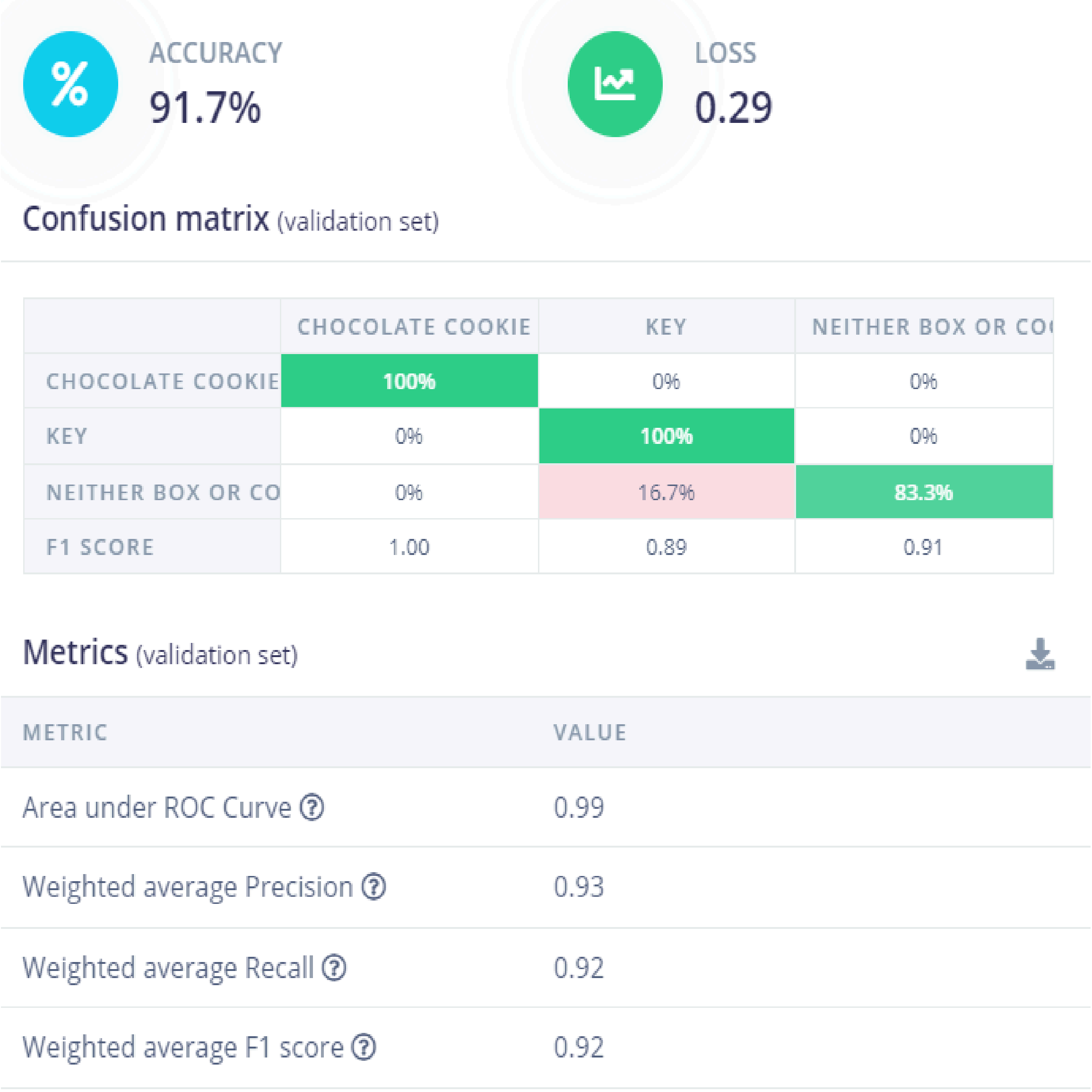
Image 3.2-3: : images data for unknown



Training set

Data in training set	60 items
Classes	3 (chocolate cookie, key, neither box or cookie)

Image 3.2-4: feature explorer for cookie, key, and unknown



ACCURACY
91.7%

LOSS
0.29

Confusion matrix (validation set)

	CHOCOLATE COOKIE	KEY	NEITHER BOX OR CO
CHOCOLATE COOKIE	100%	0%	0%
KEY	0%	100%	0%
NEITHER BOX OR CO	0%	16.7%	83.3%
F1 SCORE	1.00	0.89	0.91

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve ?	0.99
Weighted average Precision ?	0.93
Weighted average Recall ?	0.92
Weighted average F1 score ?	0.92

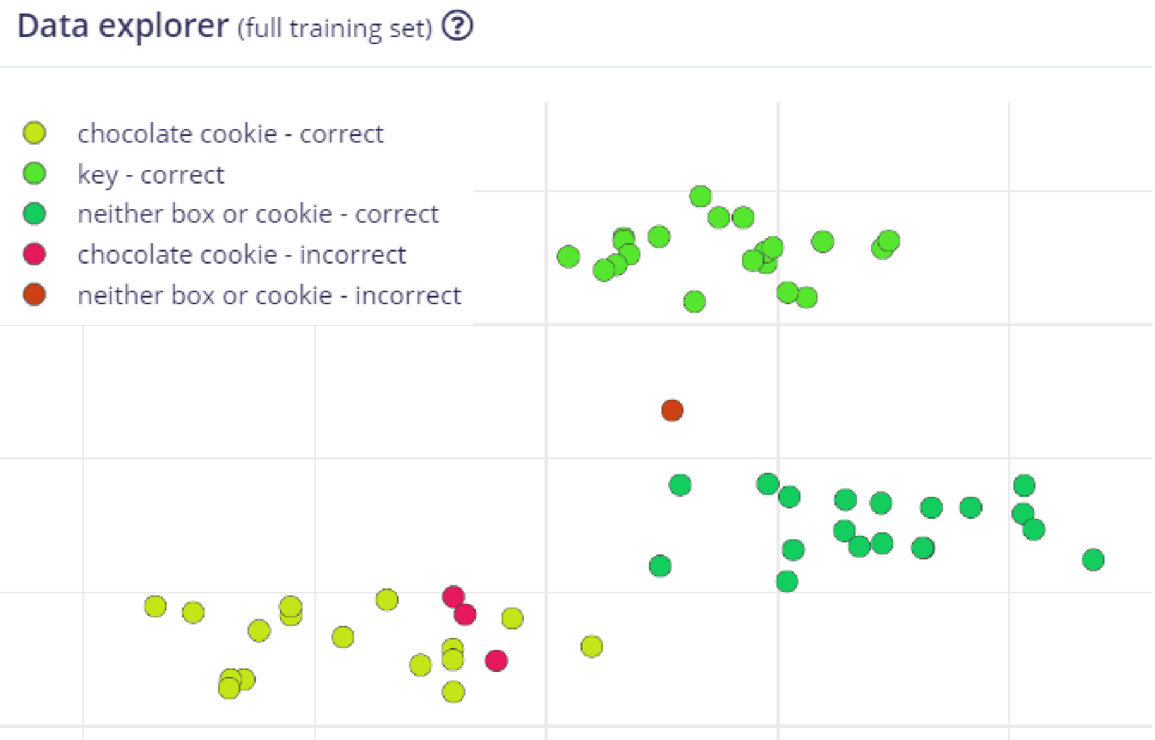
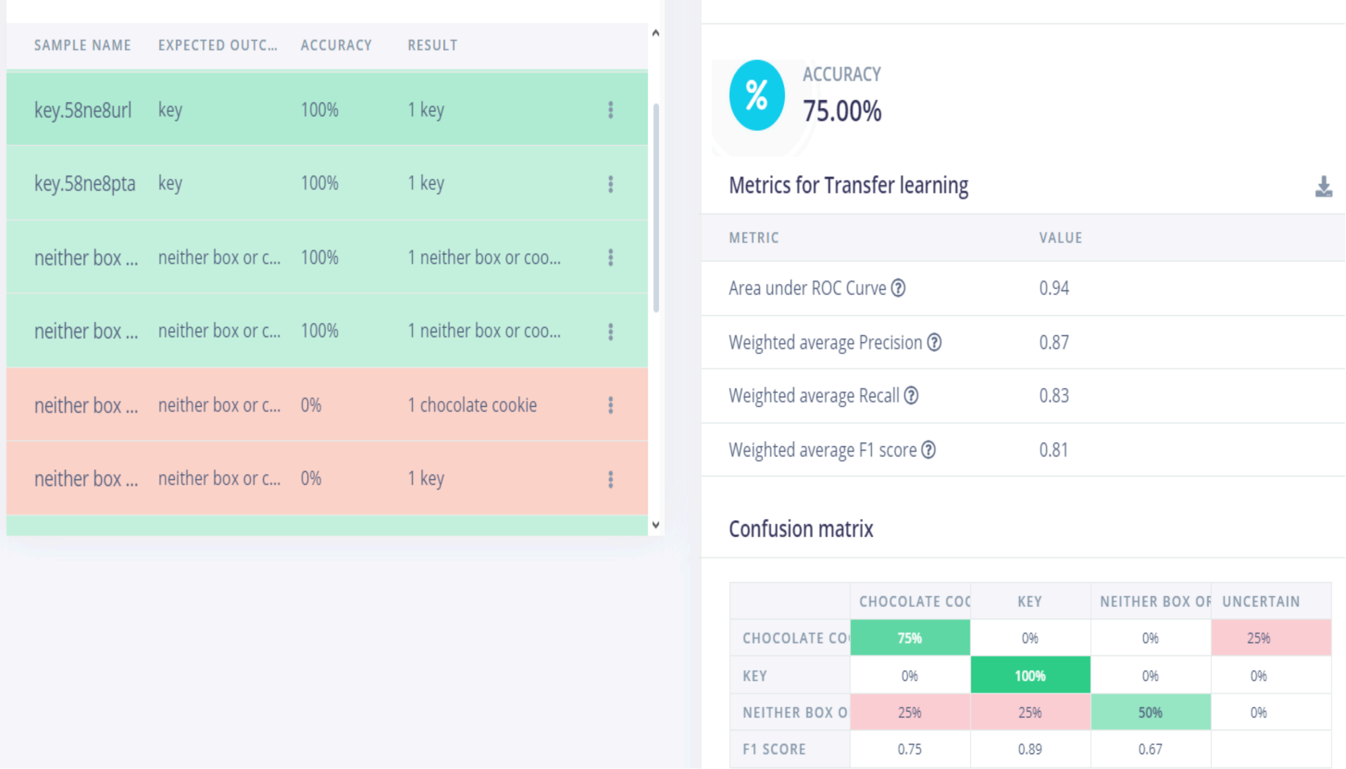


Image 3.2-5: transfer learning model



SAMPLE NAME	EXPECTED OUTC...	ACCURACY	RESULT
key.58ne8url	key	100%	1 key
key.58ne8pta	key	100%	1 key
neither box ...	neither box or c...	100%	1 neither box or coo...
neither box ...	neither box or c...	100%	1 neither box or coo...
neither box ...	neither box or c...	0%	1 chocolate cookie
neither box ...	neither box or c...	0%	1 key

ACCURACY
75.00%

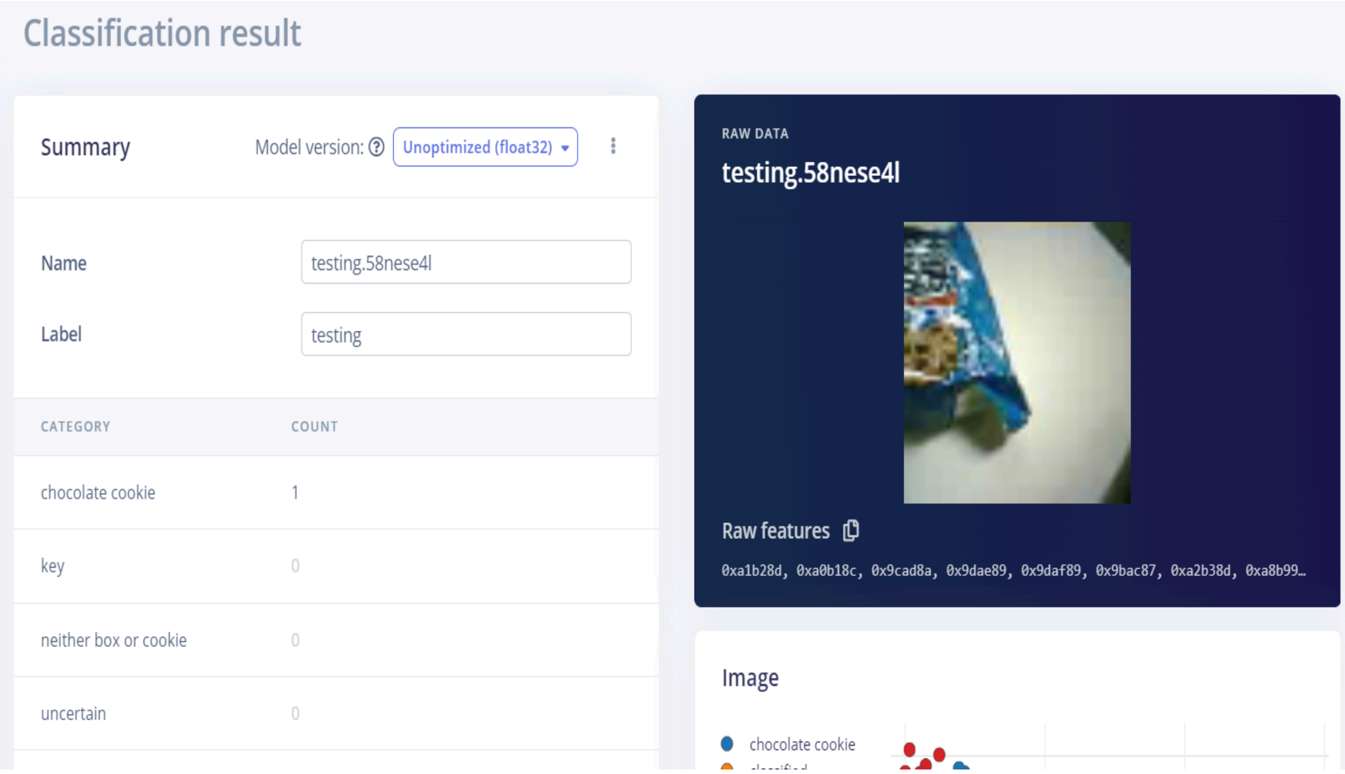
Metrics for Transfer learning

METRIC	VALUE
Area under ROC Curve ?	0.94
Weighted average Precision ?	0.87
Weighted average Recall ?	0.83
Weighted average F1 score ?	0.81

Confusion matrix

	CHOCOLATE COO	KEY	NEITHER BOX OF	UNCERTAIN
CHOCOLATE CO	75%	0%	0%	25%
KEY	0%	100%	0%	0%
NEITHER BOX O	25%	25%	50%	0%
F1 SCORE	0.75	0.89	0.67	

Image 3.2-6: Model testing



Classification result

Summary Model version: Unoptimized (float32)

Name	testing.58nese4l
Label	testing

CATEGORY	COUNT
chocolate cookie	1
key	0
neither box or cookie	0
uncertain	0

RAW DATA

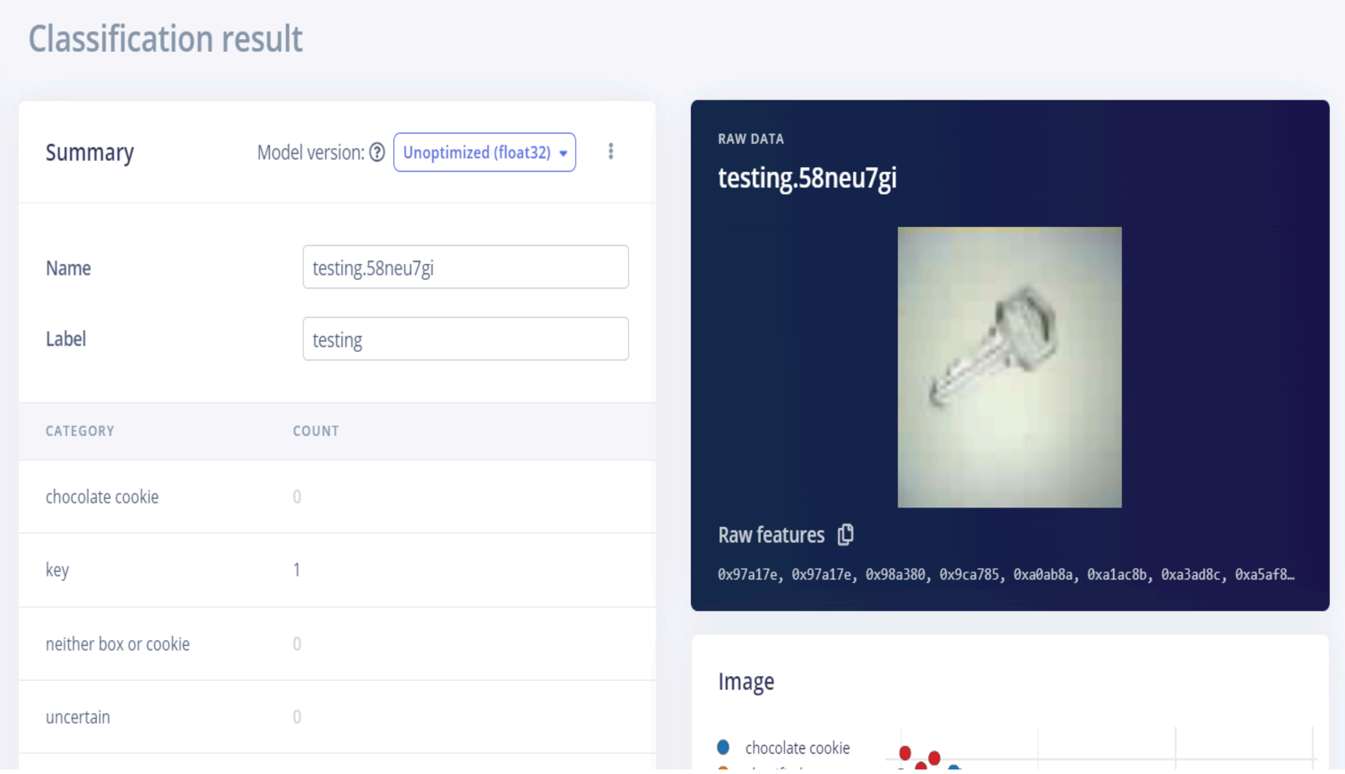
testing.58nese4l

Raw features

```
0xa1b28d, 0xa0b18c, 0x9cad8a, 0x9dae89, 0x9daf89, 0x9bac87, 0xa2b38d, 0xa8b99…
```

Image

Image 3.2-7: live classification for cookie



Classification result

Summary Model version: Unoptimized (float32)

Name	testing.58neu7gi
Label	testing

CATEGORY	COUNT
chocolate cookie	0
key	1
neither box or cookie	0
uncertain	0

RAW DATA

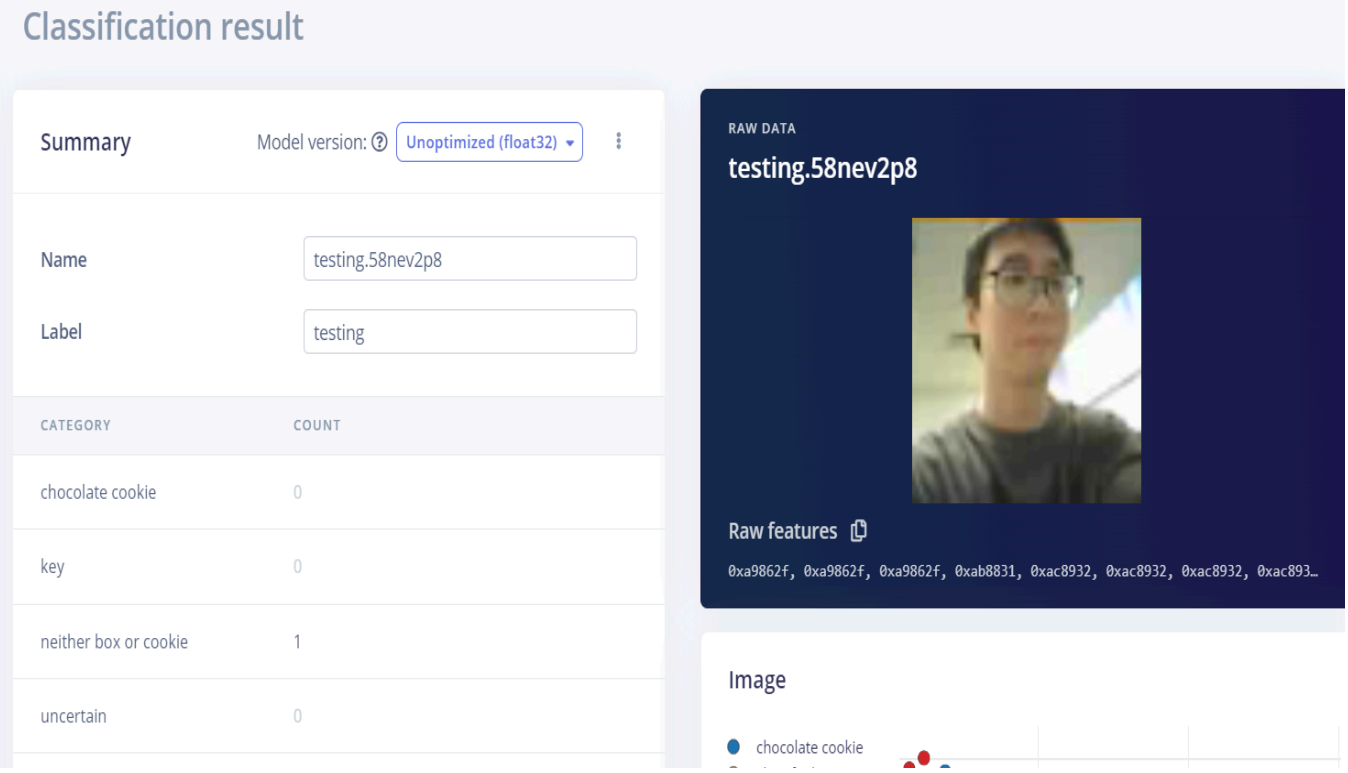
testing.58neu7gi

Raw features

```
0x97a17e, 0x97a17e, 0x98a380, 0x9ca785, 0xa0ab8a, 0xa1ac8b, 0xa3ad8c, 0xa5af8…
```

Image

Image 3.2-8: live classification for key



Classification result

Summary

Model version: Unoptimized (float32)

Name	testing.58nev2p8
Label	testing

CATEGORY	COUNT
chocolate cookie	0
key	0
neither box or cookie	1
uncertain	0

RAW DATA

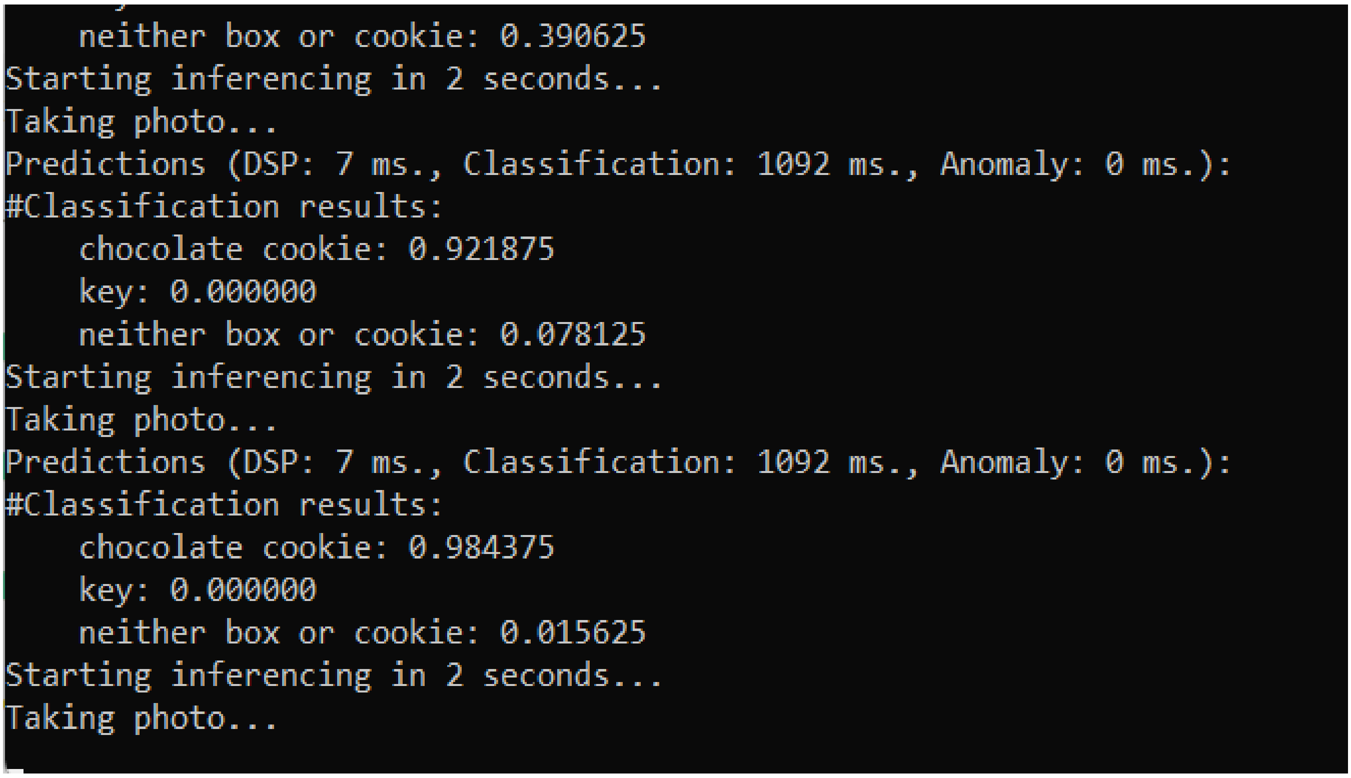
testing.58nev2p8

Raw features

0xa9862f, 0xa9862f, 0xa9862f, 0xab8831, 0xac8932, 0xac8932, 0xac8932, 0xac893…

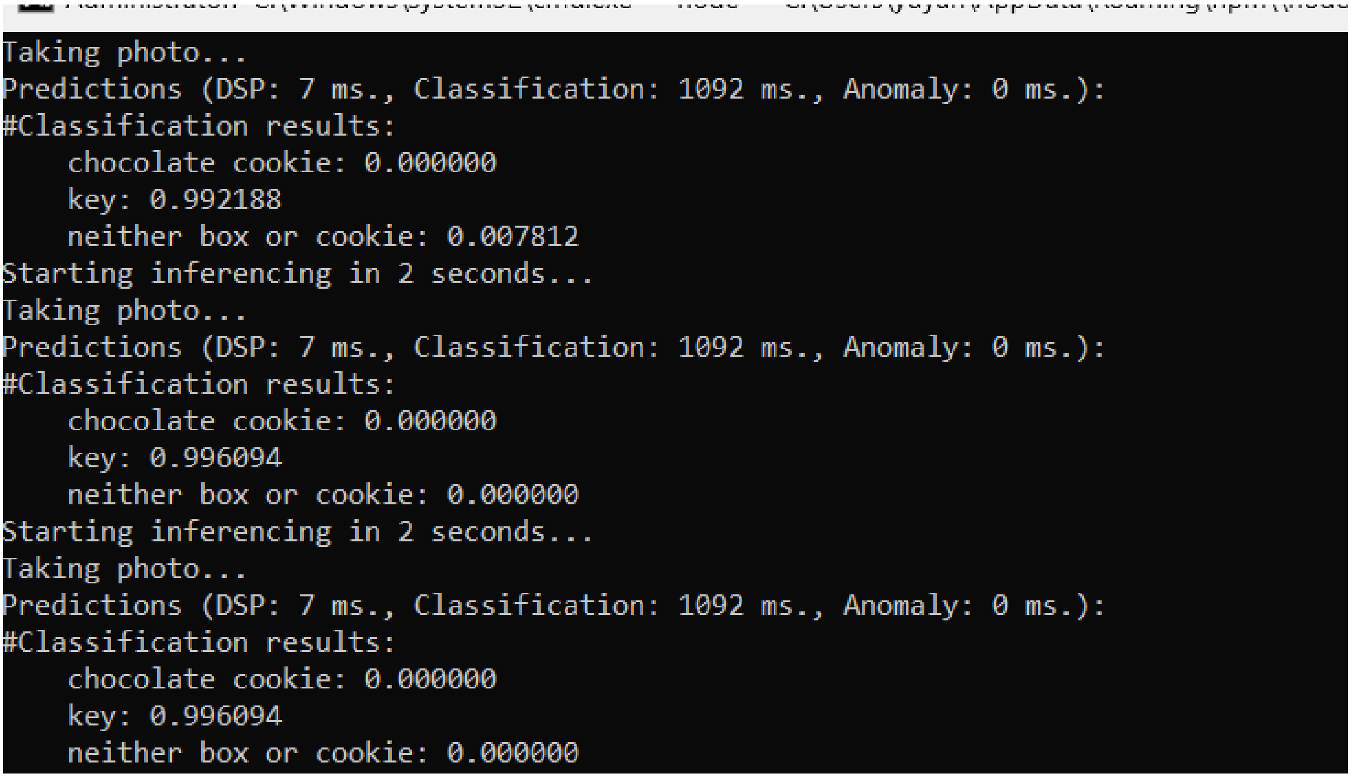
Image

Image 3.2-9: live classification for unknown



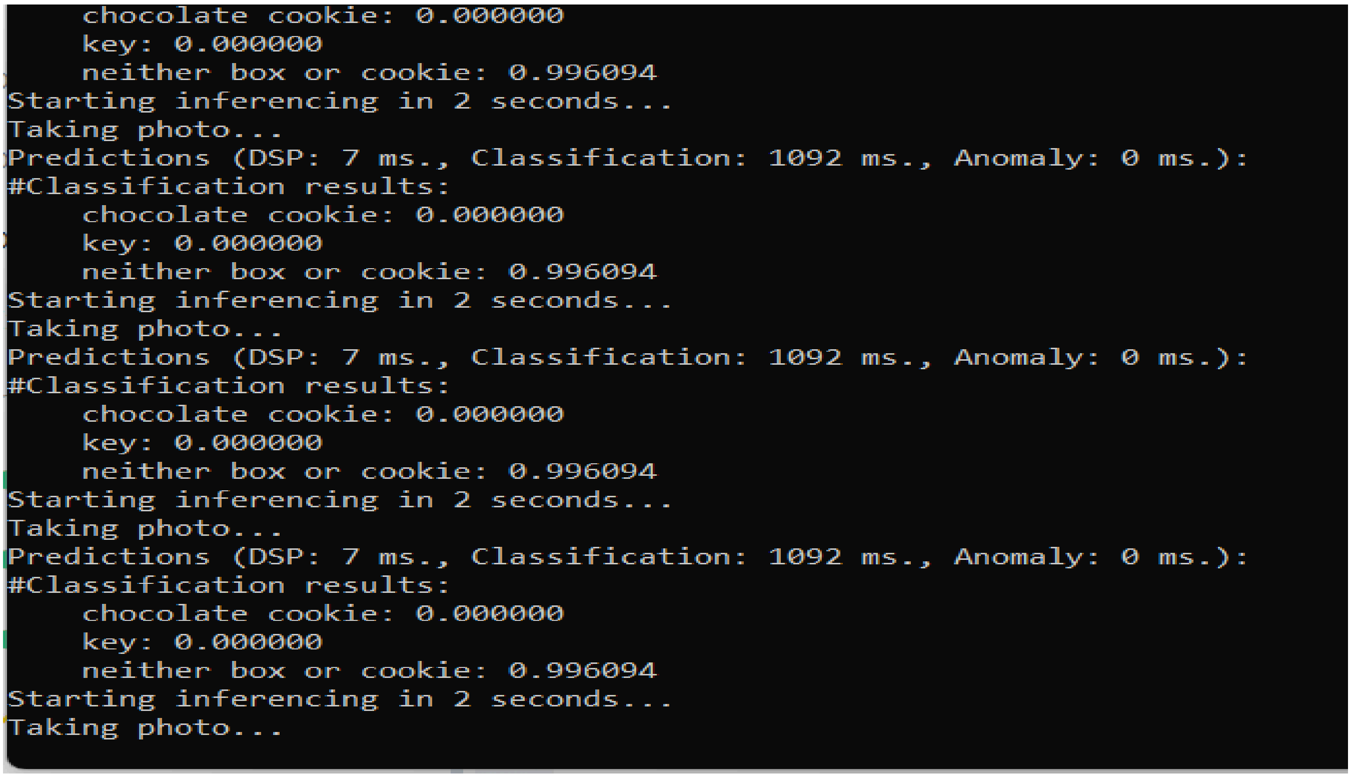
```
    neither box or cookie: 0.390625
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 7 ms., Classification: 1092 ms., Anomaly: 0 ms.):
#Classification results:
    chocolate cookie: 0.921875
    key: 0.000000
    neither box or cookie: 0.078125
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 7 ms., Classification: 1092 ms., Anomaly: 0 ms.):
#Classification results:
    chocolate cookie: 0.984375
    key: 0.000000
    neither box or cookie: 0.015625
Starting inferencing in 2 seconds...
Taking photo...
```

Image 3.2-10: output for cookie



```
Taking photo...
Predictions (DSP: 7 ms., Classification: 1092 ms., Anomaly: 0 ms.):
#Classification results:
    chocolate cookie: 0.000000
    key: 0.992188
    neither box or cookie: 0.007812
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 7 ms., Classification: 1092 ms., Anomaly: 0 ms.):
#Classification results:
    chocolate cookie: 0.000000
    key: 0.996094
    neither box or cookie: 0.000000
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 7 ms., Classification: 1092 ms., Anomaly: 0 ms.):
#Classification results:
    chocolate cookie: 0.000000
    key: 0.996094
    neither box or cookie: 0.000000
```

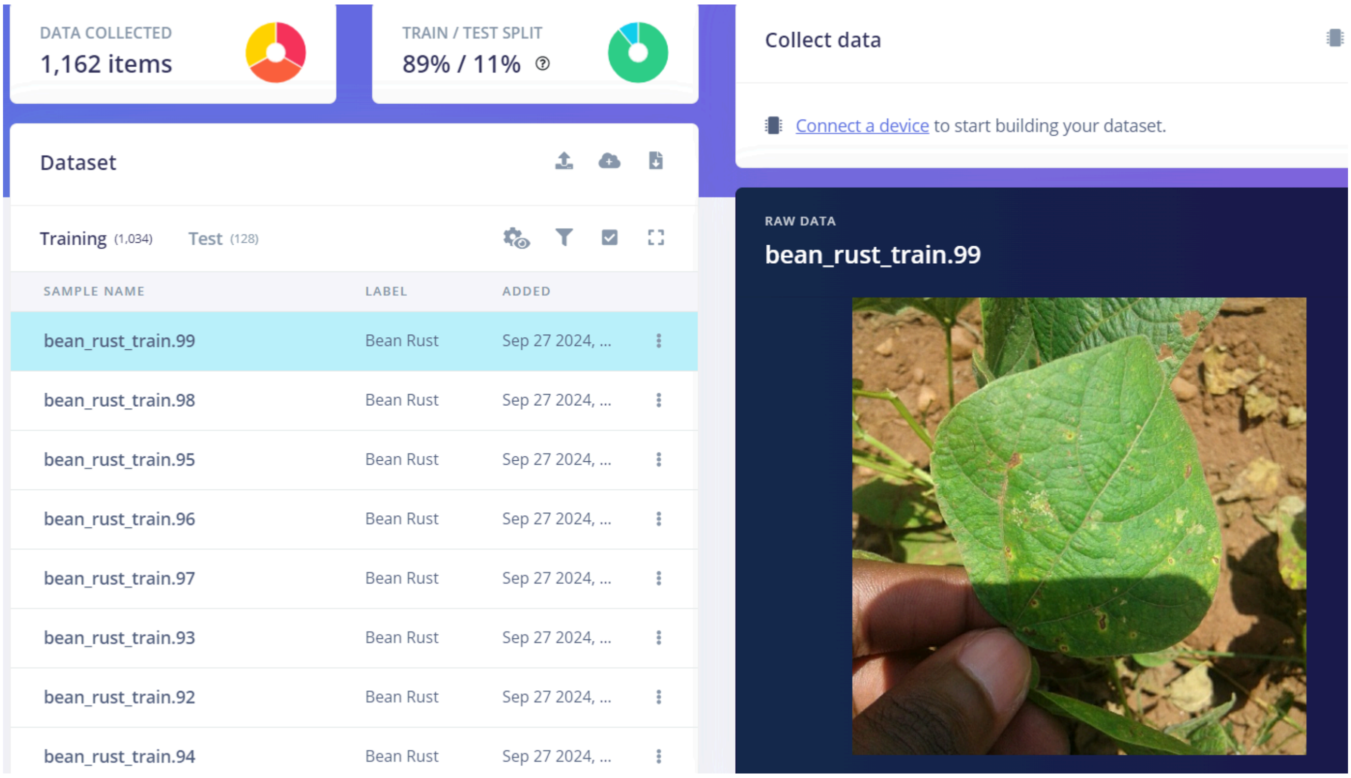
Image 3.2-11: Output for key



```
    chocolate cookie: 0.000000
    key: 0.000000
    neither box or cookie: 0.996094
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 7 ms., Classification: 1092 ms., Anomaly: 0 ms.):
#Classification results:
    chocolate cookie: 0.000000
    key: 0.000000
    neither box or cookie: 0.996094
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 7 ms., Classification: 1092 ms., Anomaly: 0 ms.):
#Classification results:
    chocolate cookie: 0.000000
    key: 0.000000
    neither box or cookie: 0.996094
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 7 ms., Classification: 1092 ms., Anomaly: 0 ms.):
#Classification results:
    chocolate cookie: 0.000000
    key: 0.000000
    neither box or cookie: 0.996094
Starting inferencing in 2 seconds...
Taking photo...
```

Image 3.2-12: Output for unknown

3.3 Experiment Images for Part 2



DATA COLLECTED
1,162 items

TRAIN / TEST SPLIT
89% / 11%

Collect data

Connect a device to start building your dataset.

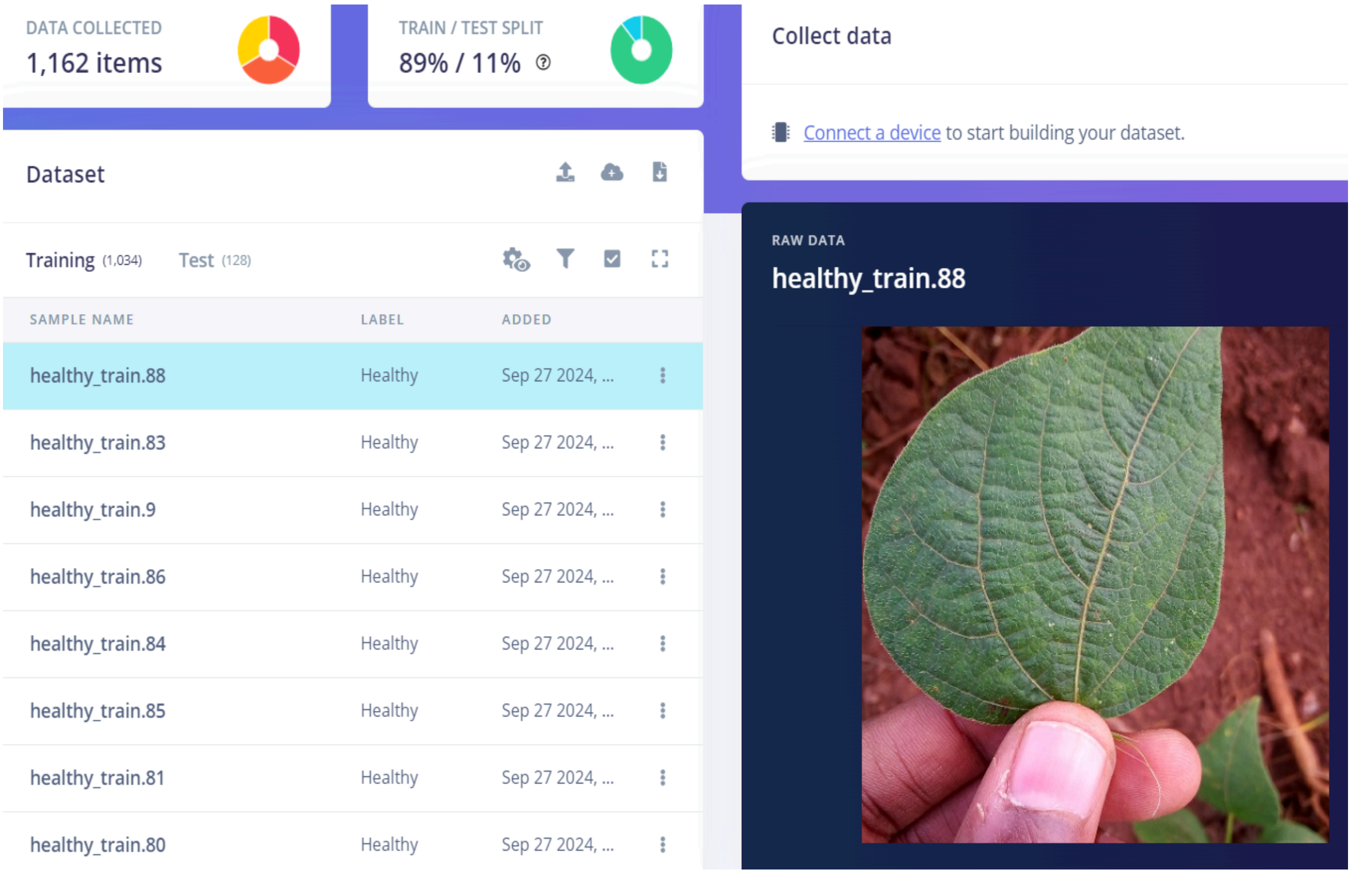
Dataset

Training (1,034) Test (128)

SAMPLE NAME	LABEL	ADDED
bean_rust_train.99	Bean Rust	Sep 27 2024, ...
bean_rust_train.98	Bean Rust	Sep 27 2024, ...
bean_rust_train.95	Bean Rust	Sep 27 2024, ...
bean_rust_train.96	Bean Rust	Sep 27 2024, ...
bean_rust_train.97	Bean Rust	Sep 27 2024, ...
bean_rust_train.93	Bean Rust	Sep 27 2024, ...
bean_rust_train.92	Bean Rust	Sep 27 2024, ...
bean_rust_train.94	Bean Rust	Sep 27 2024, ...

RAW DATA
bean_rust_train.99

Image 3.3-1: image data for bean rust



DATA COLLECTED
1,162 items

TRAIN / TEST SPLIT
89% / 11%

Collect data

Connect a device to start building your dataset.

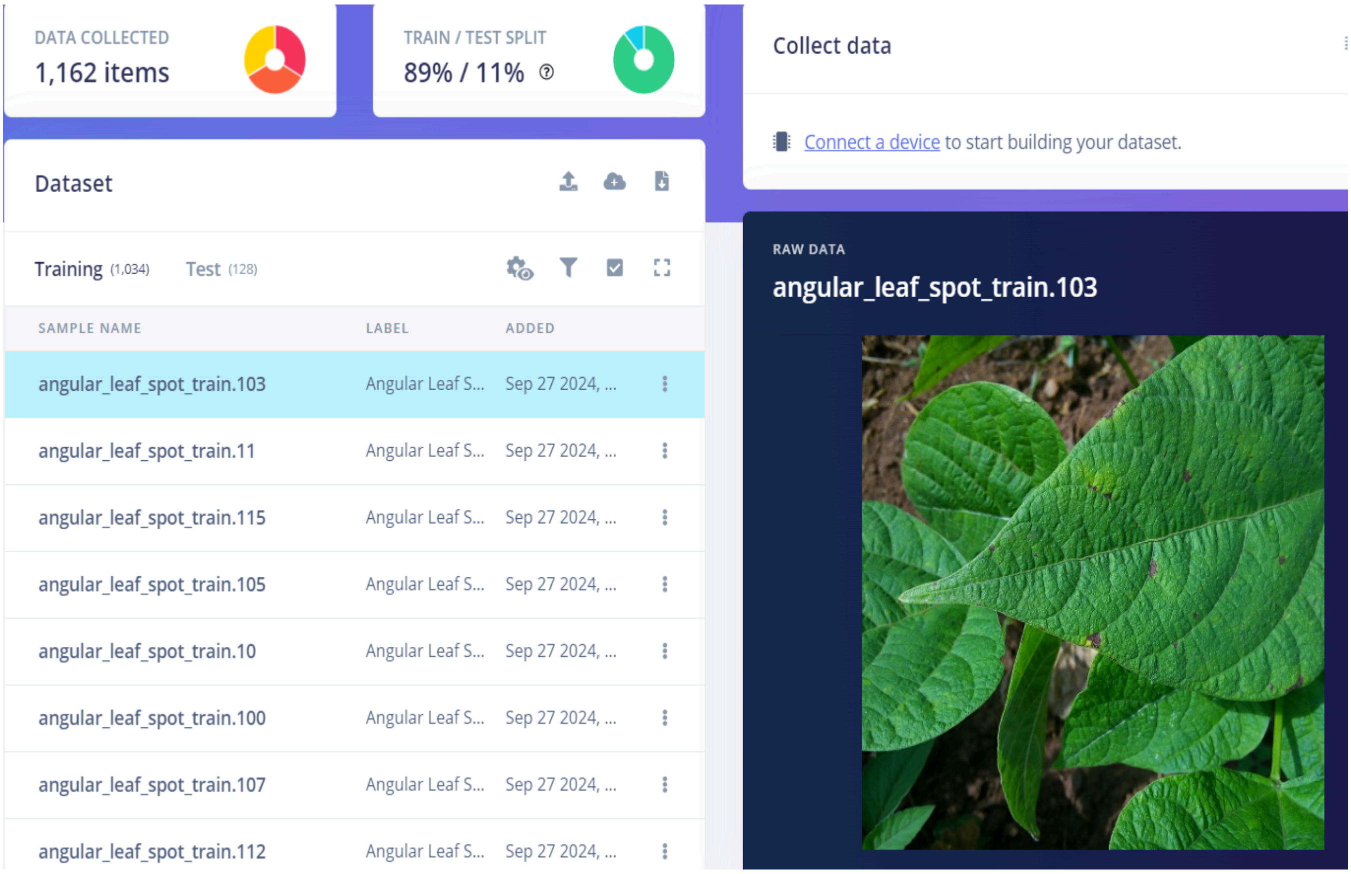
Dataset

Training (1,034) Test (128)

SAMPLE NAME	LABEL	ADDED
healthy_train.88	Healthy	Sep 27 2024, ...
healthy_train.83	Healthy	Sep 27 2024, ...
healthy_train.9	Healthy	Sep 27 2024, ...
healthy_train.86	Healthy	Sep 27 2024, ...
healthy_train.84	Healthy	Sep 27 2024, ...
healthy_train.85	Healthy	Sep 27 2024, ...
healthy_train.81	Healthy	Sep 27 2024, ...
healthy_train.80	Healthy	Sep 27 2024, ...

RAW DATA
healthy_train.88

Image 3.3-2: image data for healthy



DATA COLLECTED
1,162 items

TRAIN / TEST SPLIT
89% / 11%

Collect data

Connect a device to start building your dataset.

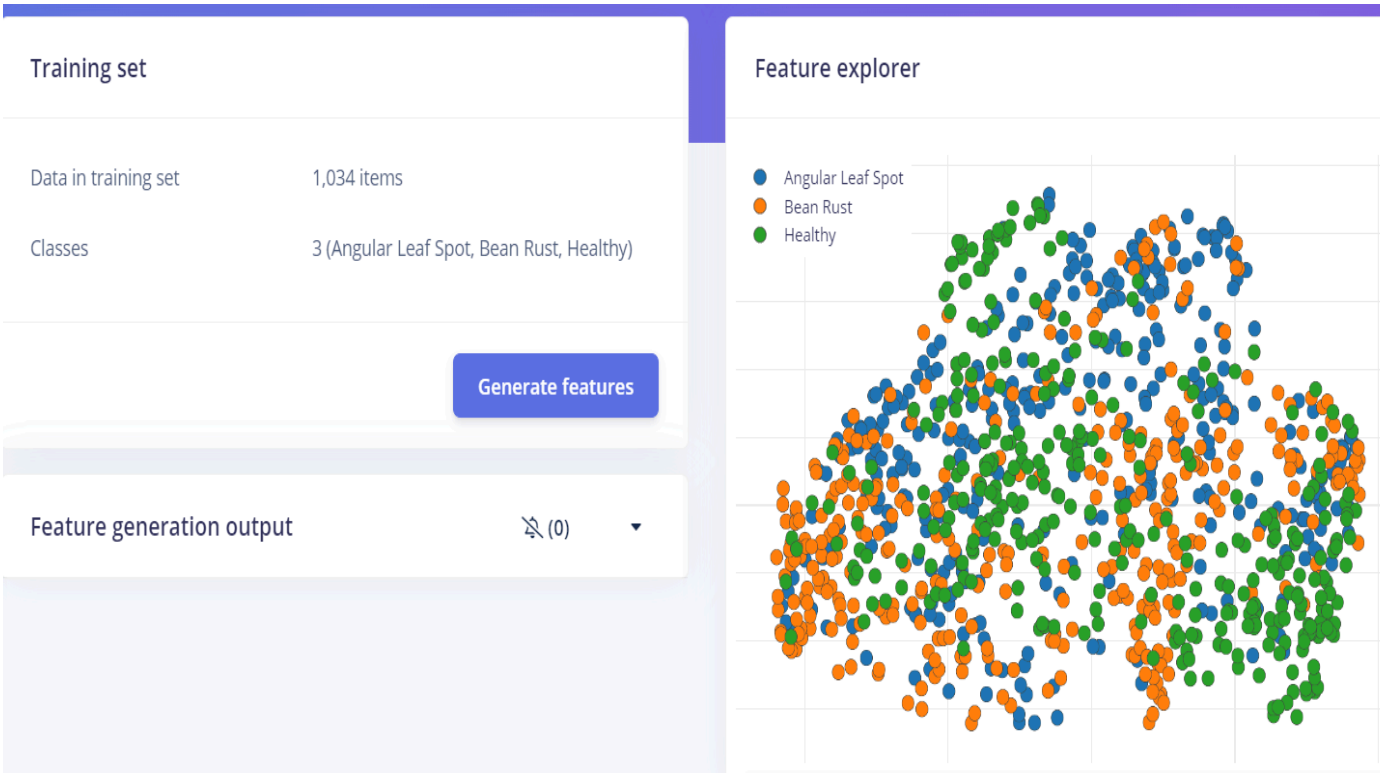
Dataset

Training (1,034) Test (128)

SAMPLE NAME	LABEL	ADDED
angular_leaf_spot_train.103	Angular Leaf S...	Sep 27 2024, ...
angular_leaf_spot_train.11	Angular Leaf S...	Sep 27 2024, ...
angular_leaf_spot_train.115	Angular Leaf S...	Sep 27 2024, ...
angular_leaf_spot_train.105	Angular Leaf S...	Sep 27 2024, ...
angular_leaf_spot_train.10	Angular Leaf S...	Sep 27 2024, ...
angular_leaf_spot_train.100	Angular Leaf S...	Sep 27 2024, ...
angular_leaf_spot_train.107	Angular Leaf S...	Sep 27 2024, ...
angular_leaf_spot_train.112	Angular Leaf S...	Sep 27 2024, ...

RAW DATA
angular_leaf_spot_train.103

Image 3.3-3: image data for angular leaf spot



Training set

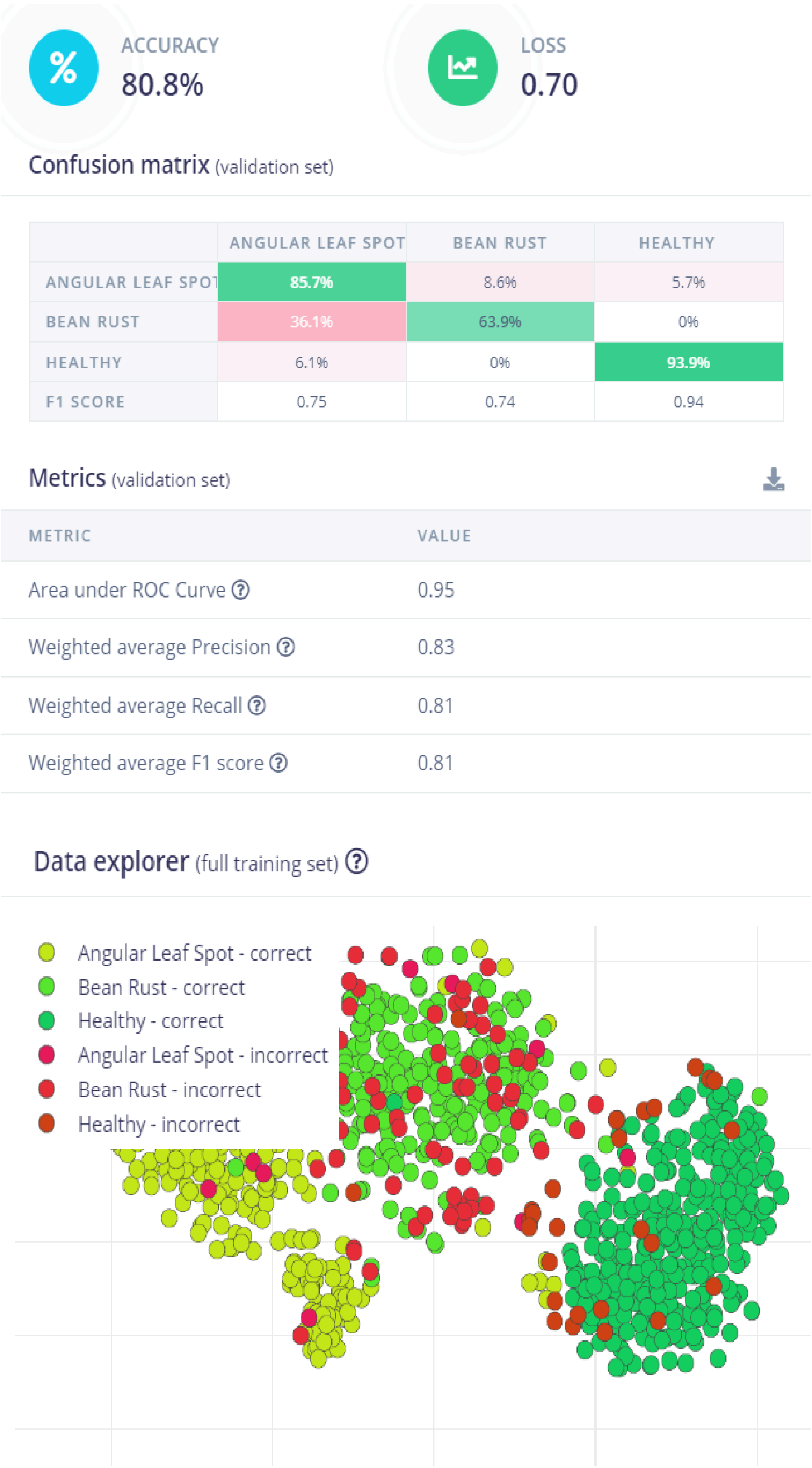
Data in training set 1,034 items

Classes 3 (Angular Leaf Spot, Bean Rust, Healthy)

Feature generation output (0)

Feature explorer

Image 3.3-4: feature explorer for angular leaf spot, bean rust ,and healthy



ACCURACY
80.8%

LOSS
0.70

Confusion matrix (validation set)

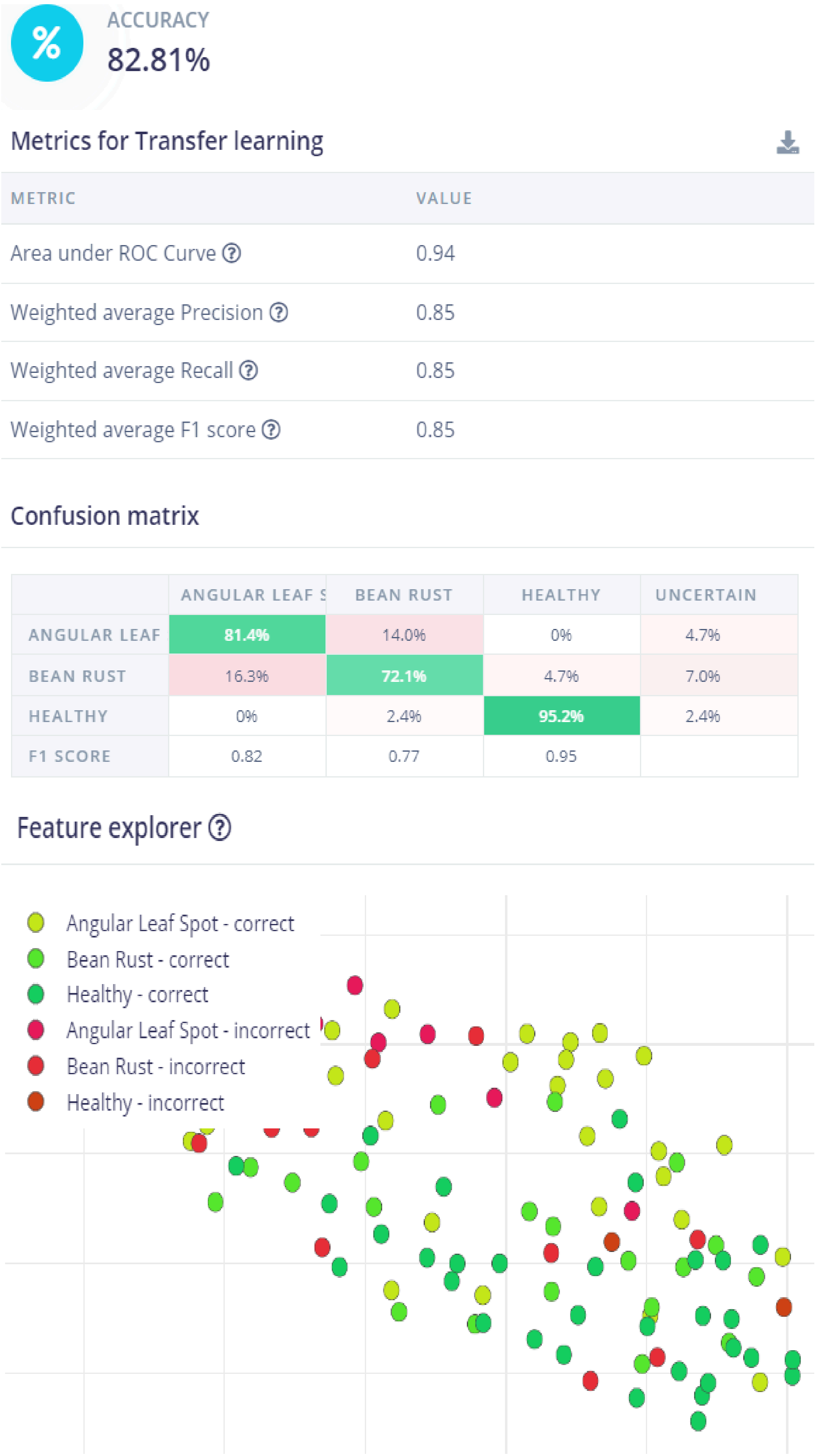
	ANGULAR LEAF SPOT	BEAN RUST	HEALTHY
ANGULAR LEAF SPOT	85.7%	8.6%	5.7%
BEAN RUST	36.1%	63.9%	0%
HEALTHY	6.1%	0%	93.9%
F1 SCORE	0.75	0.74	0.94

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve ?	0.95
Weighted average Precision ?	0.83
Weighted average Recall ?	0.81
Weighted average F1 score ?	0.81

Data explorer (full training set) ?

Image 3.3-5: transfer learning



ACCURACY

82.81%

Metrics for Transfer learning

METRIC	VALUE
Area under ROC Curve ?	0.94
Weighted average Precision ?	0.85
Weighted average Recall ?	0.85
Weighted average F1 score ?	0.85

Confusion matrix

	ANGULAR LEAF S	BEAN RUST	HEALTHY	UNCERTAIN
ANGULAR LEAF	81.4%	14.0%	0%	4.7%
BEAN RUST	16.3%	72.1%	4.7%	7.0%
HEALTHY	0%	2.4%	95.2%	2.4%
F1 SCORE	0.82	0.77	0.95	

Feature explorer ?

Image 3.3-6: model testing

4. Discussions and Conclusions:

4.1 Comparison of your experimental results with the research/theory of the concept with reasoning.

This experiment compared the process of image classification using a microcontroller and a camera with established theories in machine learning, especially those dealing with transfer learning and neural networks. The results of the experiment were the classification of images to recognize objects, where the accuracy was presented consistently under the expectations that were outlined by theory. In particular, the transfer learning proved quite useful for allowing the model to generalize well, even on a limited dataset. It also agrees with the theoretical understanding that transfer learning works on pre-trained models in adapting quickly to new tasks, hence reducing the number of large training datasets and computationally extensive resources.

The practical application of neural networks in the experiment followed the progressions that were expected from research regarding training cycles, where model accuracy improved with time, and overfitting was minimized by the use of a validation set and proper tuning of the model. Theoretically, methods such as the adjustment of learning rates, introduction of dropout layers, and data augmentation help in the prevention of overfitting; this was indeed observed as the trained model generalized well on unseen test data.

The same has been further supported with the application of Hugging Face Hub in managing datasets and the creation of models, since theory suggests that access to high-quality datasets and prebuilt models enhances model performance. An experiment confirmed the advantage of using open-source platforms to speed up development while accomplishing accurate and reliable results. These are instances showing that practical outcomes in this lab correspond with theoretical machine learning frameworks.

4.2 Learnings from the experiment

This lab involved building a machine learning system capable of performing image classification using a microcontroller connected to a camera. The goal was to enable embedded devices to recognize objects, with potential applications like differentiating between poachers and elephants, automating quality control in factories, or enabling autonomous driving in RC cars. The process included collecting a well-balanced dataset of images, applying transfer learning to train a neural network, and deploying the model onto an embedded device for real-time image classification.

Additionally, the lab introduced the Hugging Face Hub, an open-source platform offering an extensive collection of datasets and machine learning models. This platform fosters collaboration within the machine learning community, allowing developers to build and share AI-driven projects. By utilizing the tools and resources from Hugging Face, machine learning models were explored and experimented with to enhance the capabilities of embedded systems.

5. References:

Hugging Face Hub documentation: https://huggingface.co/docs/hub/index
Beans dataset from HuggingFace Datasets: https://huggingface.co/datasets/AI-Lab-Makerere/beans
Espressif ESP-EYE (ESP32) data sets:
https://github.com/espressif/esp-who/blob/master/docs/en/get-started/ESP-EYE_Getting_Started_Guide.md

Edge Impulse: https://edgeimpulse.com/