CS249r Fall 2023 Assignment 1: Setup Arduino Nicla Vision + Data Engineering for a Person Detection Transfer Learning Model

Part 1 Due: Monday Sep 25, 2023 at 11:59pm on Canvas
Part 2 Due: Monday Oct 2, 2023 at 11:59pm on Canvas

Assignment Overview

The purpose of this assignment is to get familiar with the Arduino Nicla Vision and the general workflow for designing and deploying a TinyML application. For this assignment, you can use open source data, web scraped data, synthetic data, or manually generated data. Our goal here is to get you to explore the effect of data quality on the performance of a model, as well as the quantity of data and other factors.

You need to collect 100 samples for the **"person"** class which is our target class and an additional 100 samples for the **"background/no-person"** class. This will constitute the negative class in a binary classification setting.

This assignment is split into two parts:

- In **Part 1**:
    - **What**: You will **set up the Arduino Nicla Vision** and submit an image of yourself taken with the Nicla. Your image will be included in the testing dataset for Part 2.
    - **Why:** This is to test your board and make sure it is working and get you familiar with some open source toolchains for working with embedded systems.
    - **How:** You will use the Edge Impulse pipeline to bootstrap your system.

- In **Part 2**:
    - **What:**You will **gather training data for a person detection model**.
    - **Why:** The goal of Part 2 is to explore good data engineering techniques for training a TinyML model. We will provide you with information about the model but will hide the testing data which will be used to assess the quality of your training data. The testing data contains 500 images, including photos submitted by students taking this course.
    - **How:** You will use Edge Impulse + OpenMV tools to get your training data.

Please ask questions (and if you can, answer your fellow classmates' questions) in #assignment1 in our Slack workspace. We also have office hours at different times in the week listed in the Course Syllabus.

Deliverables

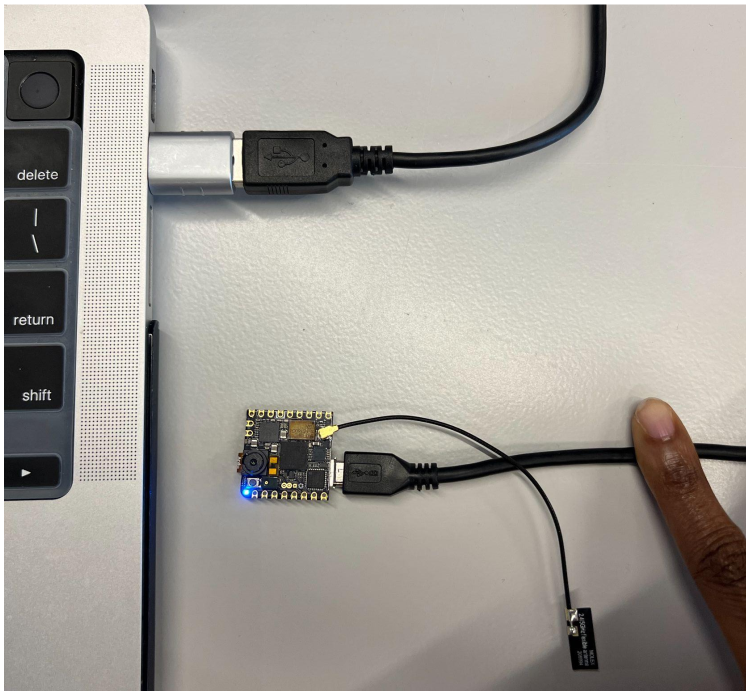
Part 1: Submit a single zip file containing
i) **an .mp4 video of length at most 2 mins showing your Nicla** connected to Edge Impulse and capable of sampling images of your face
ii) **a 96x96 image of yourself** for the assignment testing data

Part 2: Submit a single zip file containing
i) a **training dataset** made up of a total of 200 images that belong to a ***"person"*** and ***"non-person"*** category. This dataset will be used to train MobileNetV2 96x96 0.1 transfer learning model in Edge Impulse.
ii) **a PDF write up** of your dataset design decisions.

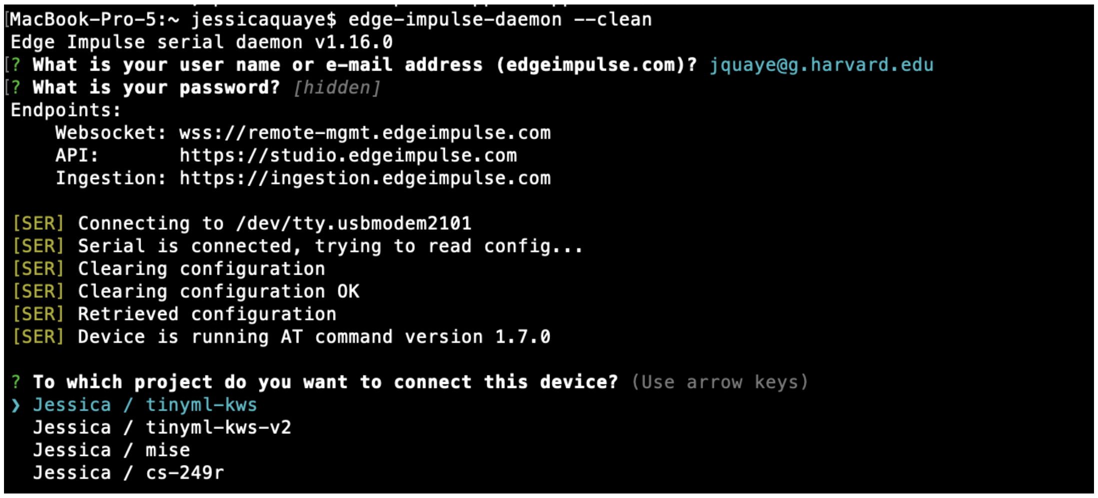
For each submission, create a zip file containing the requested documents and upload it to canvas. You can collaborate with others on setting up your kit and using the Edge Impulse platform, **but you will need to collect your own dataset and complete each of the above deliverables individually**.

Instructions for Part 1

1. Connect your Arduino Nicla Vision to your laptop using the micro-USB to USB-A cable that came with your kit.

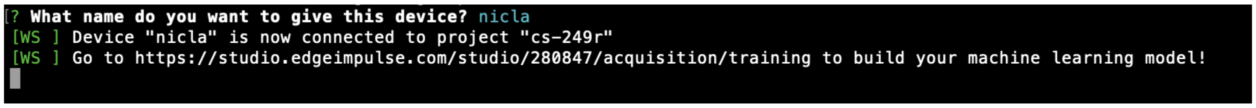


2. Install all the dependencies required to set up your Arduino Nicla Vision device on your machine.
   After installing the dependencies listed on the website linked above, you should be able to connect your Arduino Nicla to the Edge Impulse portal via command line.
3. Follow the steps below to take your picture:
   a. Create a project in Edge Impulse.
   b. In your terminal, run “edge-impulse-daemon”. When prompted, provide your login details and select the project which you want to connect the device to from the dropdown menu (if you have multiple projects in Edge Impulse)



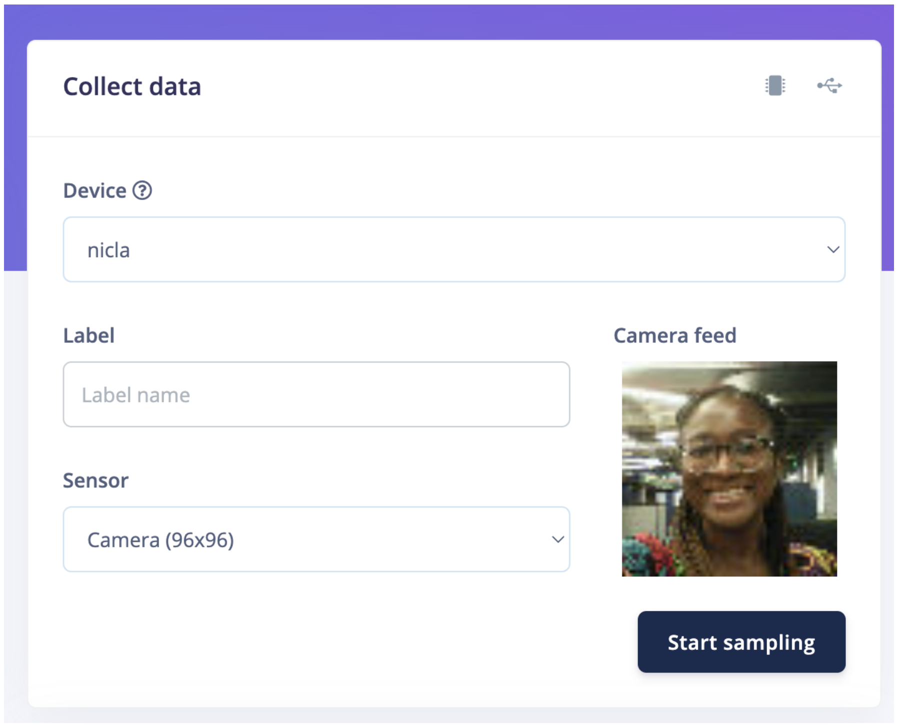
```
MacBook-Pro-5:~ jessicaquaye$ edge-impulse-daemon --clean
Edge Impulse serial daemon v1.16.0
? What is your user name or e-mail address (edgeimpulse.com)? jquaye@g.harvard.edu
? What is your password? [hidden]
Endpoints:
    Websocket: wss://remote-mgmt.edgeimpulse.com
    API:       https://studio.edgeimpulse.com
    Ingestion: https://ingestion.edgeimpulse.com

[SER] Connecting to /dev/tty.usbmodem2101
[SER] Serial is connected, trying to read config...
[SER] Clearing configuration
[SER] Clearing configuration OK
[SER] Retrieved configuration
[SER] Device is running AT command version 1.7.0

? To which project do you want to connect this device? (Use arrow keys)
❯ Jessica / tinyml-kws
  Jessica / tinyml-kws-v2
  Jessica / mise
  Jessica / cs-249r
```

4. Name the device so that you can reference it in the Edge Impulse portal:



```
? What name do you want to give this device? nicla
[WS ] Device "nicla" is now connected to project "cs-249r"
[WS ] Go to https://studio.edgeimpulse.com/studio/280847/acquisition/training to build your machine learning model!
```

5. Once your device is named and connected to the project, you can start collecting data! Click on *Data Acquisition* from the left-hand navigation panel on the Edge Impulse website to view the real-time camera live stream in your web browser. In the “Collect Data” section, select your device (it will be listed as the name that you gave it in Step 4) and choose “Camera (96x96)” as the sensor:



6. Download your picture from the Edge Impulse platform for submission.

Part 1 Deliverables
Submit a .zip file on Canvas containing

1. A video recording of your Nicla connected to Edge Impulse and streaming from the camera.
2. A 96x96 RGB image of yourself to be used in the testing dataset.

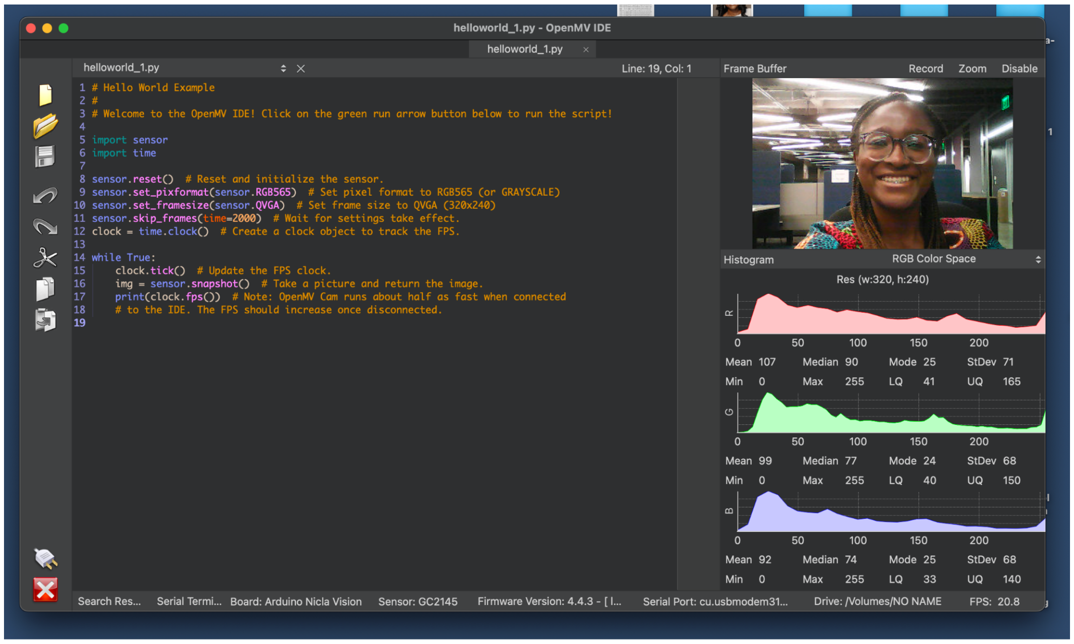
Instructions for Part 2

If you are comfortable using the Edge Impulse interface, you do not have to complete this step. However, the data collection process can be simplified by using the OpenMV IDE:

1. Download the OpenMV IDE.
   NOTE: Macbook Users might have to download this Github release to avoid the GUI from crashing when attempting to connect the Nicla to the OpenMV IDE.

2. Connect the Arduino Nicla to OpenMV IDE.

3. Once your Nicla is connected, navigate to File > Examples > HelloWorld > helloworld.py and click on the green button at the bottom left to Run the program. You should be able to see a real-time camera live stream in the Frame Buffer on the right.

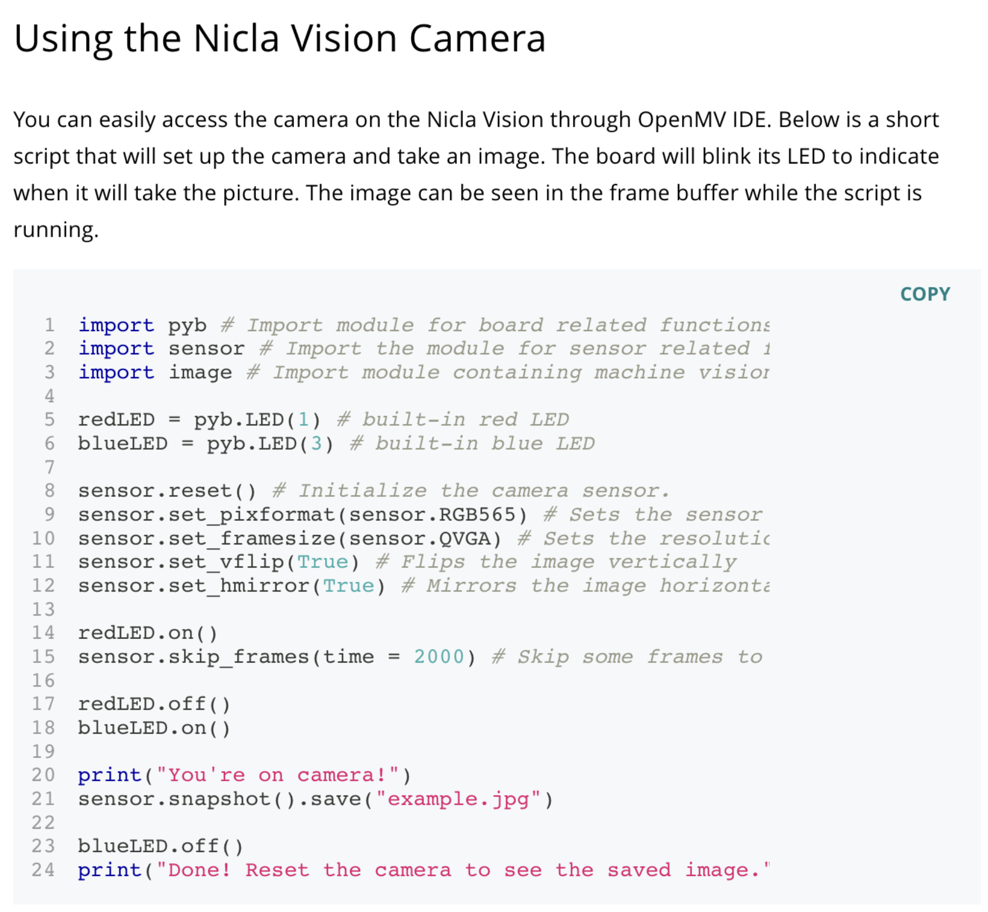


```
# Hello World Example
#
# Welcome to the OpenMV IDE! Click on the green run arrow button below to run the script!

import sensor
import time

sensor.reset()  # Reset and initialize the sensor.
sensor.set_pixformat(sensor.RGB565)  # Set pixel format to RGB565 (or GRAYSCALE)
sensor.set_framesize(sensor.QVGA)  # Set frame size to QVGA (320x240)
sensor.skip_frames(time=2000)  # Wait for settings take effect.
clock = time.clock()  # Create a clock object to track the FPS.

while True:
    clock.tick()  # Update the FPS clock.
    img = sensor.snapshot()  # Take a picture and return the image.
    print(clock.fps())  # Note: OpenMV Cam runs about half as fast when connected
    # to the IDE. The FPS should increase once disconnected.
```

```
Mean  107    Median  90     Mode  25    StDev  71
Min   0      Max     255    LQ    41    UQ     165

Mean  99     Median  77     Mode  24    StDev  68
Min   0      Max     255    LQ    40    UQ     150

Mean  92     Median  74     Mode  25    StDev  68
Min   0      Max     255    LQ    33    UQ     140
```

Search Res...   Serial Termi...   Board: Arduino Nicla Vision   Sensor: GC2145   Firmware Version: 4.4.3 - [l...   Serial Port: cu.usbmodem31...   Drive: /Volumes/NO NAME   FPS: 20.8

4. Use the code below from the Nicla website to take pictures which will be stored in the path defined in line 21. You can then upload these pictures into the Edge Impulse platform for training.



Using the Nicla Vision Camera

You can easily access the camera on the Nicla Vision through OpenMV IDE. Below is a short script that will set up the camera and take an image. The board will blink its LED to indicate when it will take the picture. The image can be seen in the frame buffer while the script is running.

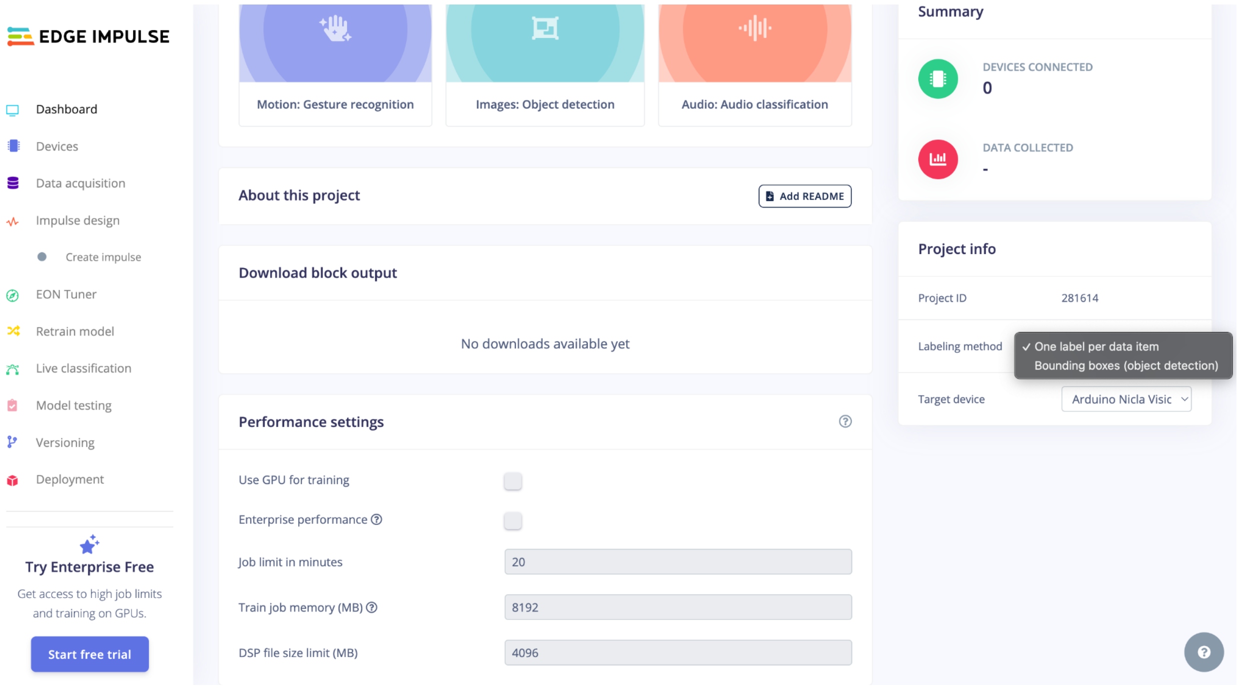
```python
import pyb # Import module for board related functions
import sensor # Import the module for sensor related i
import image # Import module containing machine visior

redLED = pyb.LED(1) # built-in red LED
blueLED = pyb.LED(3) # built-in blue LED

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.RGB565) # Sets the sensor
sensor.set_framesize(sensor.QVGA) # Sets the resolutic
sensor.set_vflip(True) # Flips the image vertically
sensor.set_hmirror(True) # Mirrors the image horizonta

redLED.on()
sensor.skip_frames(time = 2000) # Skip some frames to

redLED.off()
blueLED.on()

print("You're on camera!")
sensor.snapshot().save("example.jpg")

blueLED.off()
print("Done! Reset the camera to see the saved image."
```

Training and Assessing your Model Performance on Edge Impulse

In this section, you will create a training dataset for the MobileNetV2 96x96 0.1 transfer learning model. To help with the process of creating the Machine Learning model, we will be using EdgeImpulse Studio. This contains a suite of tools used to create the model and generate a C++ library that can be used for inference and image classification.

Step 1: Edge Impulse Project Setup

1. Once you have created an account, log in and create a new project. This will open the project setup dialogue where you can set the project name.
2. The project will open in the dashboard view with a menu in the left-hand margin for setting up the workflow. Scroll down and edit the Project Info on the bottom right as follows:
    a. Labeling Method: One label per data item
    b. Target Device: Arduino Nicla Vision



Use GPU for training
Enterprise performance
Job limit in minutes 20
Train job memory (MB) 8192
DSP file size limit (MB) 4096

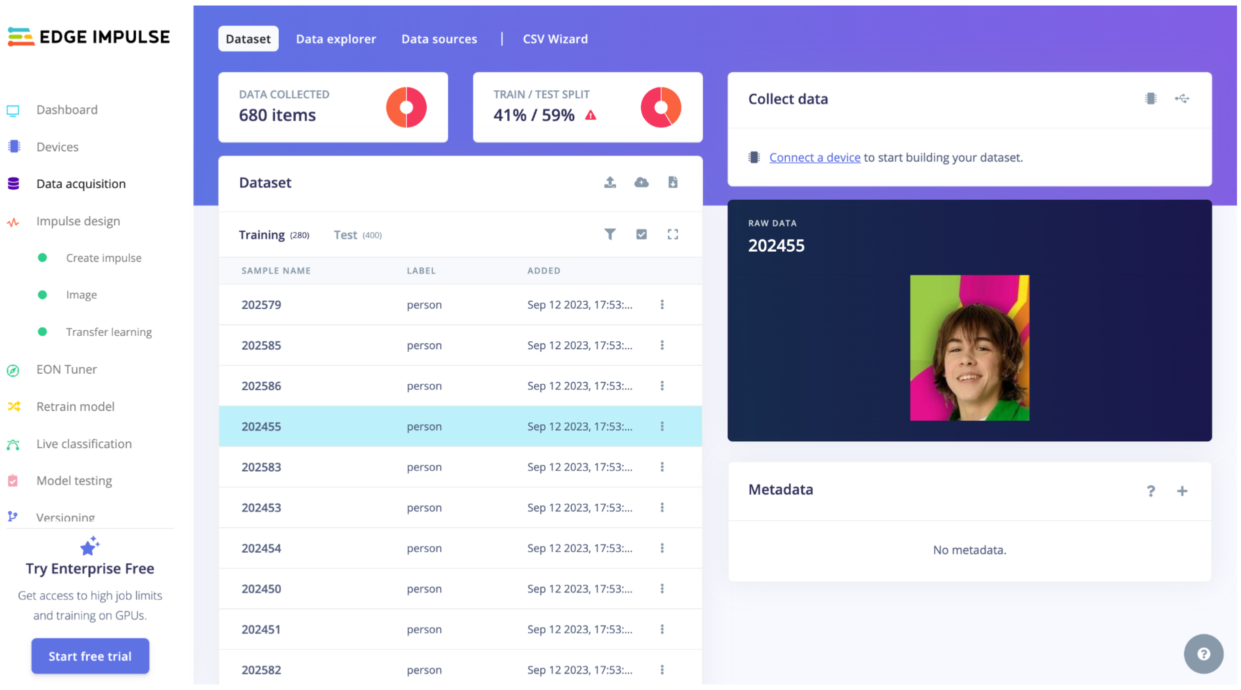
DEVICES CONNECTED 0
DATA COLLECTED -

Project info

Project ID	281614
Labeling method	✓ One label per data item Bounding boxes (object detection)
Target device	Arduino Nicla Visic

Step 2: Data Acquisition

1. Navigate to the project that you created for this assignment in EdgeImpulse.
2. Click the ***Data Acquisition*** activity from the left-hand navigation panel.
3. Click ***Choose Files*** and select the images from the relevant folder on your laptop.
4. Give the images a label (“person” or “non-person”, depending on which class of images you’re uploading) and begin the upload.
5. Repeat the process for each of your data classes.
6. You should now have all your data loaded into EdgeImpulse ready for processing. You can either upload your training dataset and your testing dataset separately or ask for EdgeImpulse to automatically split the data that you upload into 2 parts.



DATA COLLECTED 680 items

TRAIN / TEST SPLIT 41% / 59%

Collect data

Connect a device to start building your dataset.

Dataset

Training (280) Test (400)

SAMPLE NAME	LABEL	ADDED
202579	person	Sep 12 2023, 17:53:...
202585	person	Sep 12 2023, 17:53:...
202586	person	Sep 12 2023, 17:53:...
202455	person	Sep 12 2023, 17:53:...
202583	person	Sep 12 2023, 17:53:...
202453	person	Sep 12 2023, 17:53:...
202454	person	Sep 12 2023, 17:53:...
202450	person	Sep 12 2023, 17:53:...
202451	person	Sep 12 2023, 17:53:...
202582	person	Sep 12 2023, 17:53:...

RAW DATA
202455

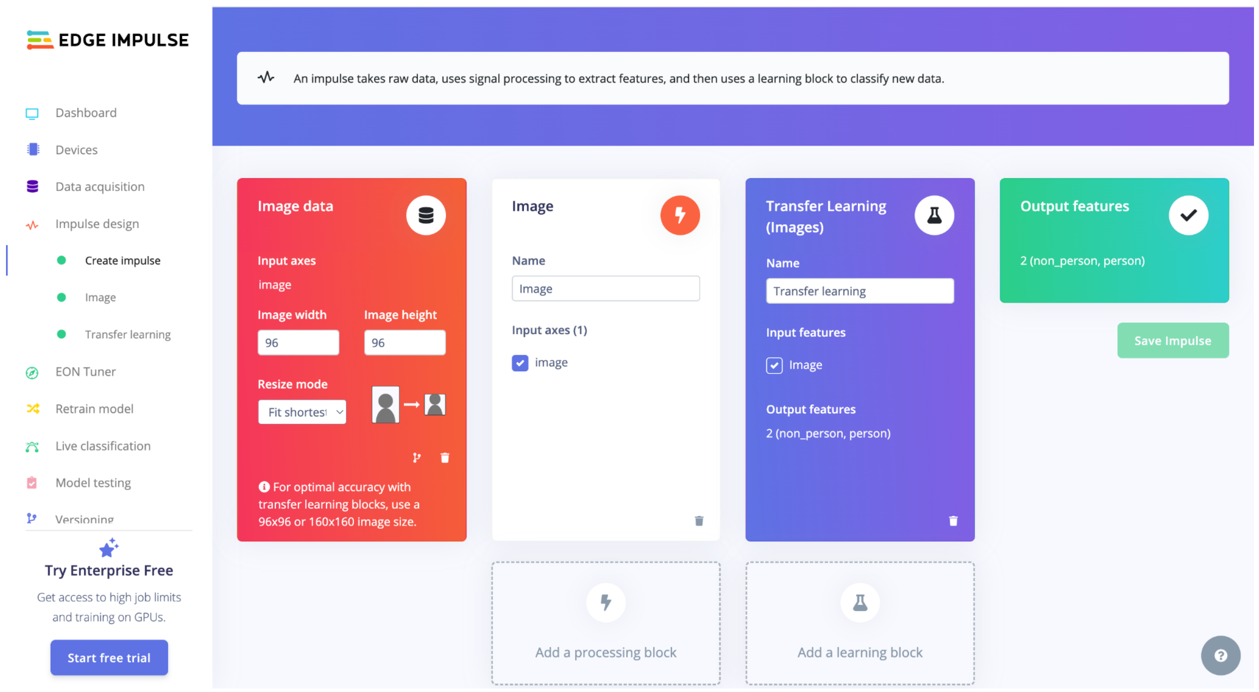
Metadata

No metadata.

Step 3: Impulse Design
This step sets up all the parameters for building the ML model. EdgeImpulse guides you through this phase and indicates how accurate your model is likely to be after training.

The first step to designing an impulse is to **Create Impulse**.

- Click the *Create Impulse* activity in the menu bar.
- Add an image-processing block (*Image*) and set the image size to 96x96.
- Add a *Transfer Learning* block.
- There should be two *Output Features* (person, not_person) automatically populated if your training data was labeled correctly.
- Click Save Impulse.



An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.

Image data
Input axes: image
Image width: 96
Image height: 96
Resize mode: Fit shortest
For optimal accuracy with transfer learning blocks, use a 96x96 or 160x160 image size.

Image
Name: Image
Input axes (1): ✓ image

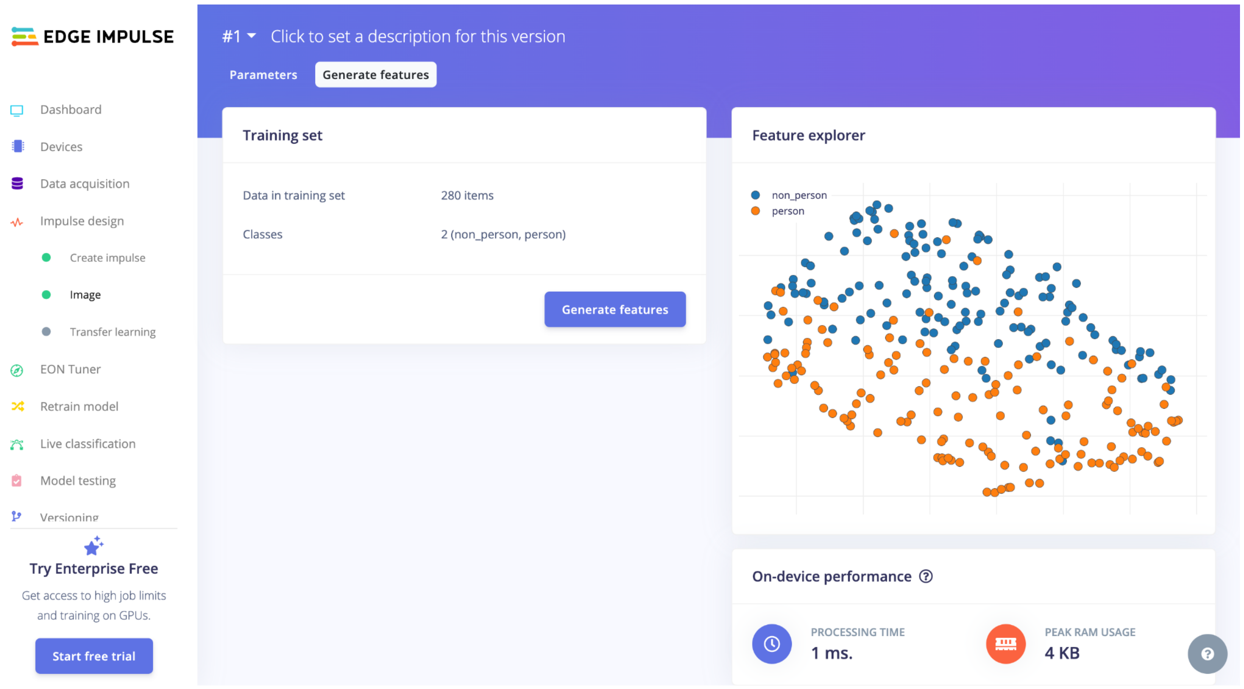
Transfer Learning (Images)
Name: Transfer learning
Input features: ✓ Image
Output features: 2 (non_person, person)

Output features
2 (non_person, person)

The next stage, **Image**, sets the model parameters and generates a feature set from the training data images.

- Click the *Image* Item from the menu bar.
- Set the *Color Depth* to RGB.
- Click *Save parameters*.
- Then, select *Generate Features*. This will trigger a remote job in the EdgeImpulse Cloud to generate the model features.

At this stage, you can use the *Feature Explorer* to gauge whether the features are sufficiently distinct to give good results. The goal is to look for distinct clusters of features with as little overlap as possible.



Data in training set	280 items
Classes	2 (non_person, person)

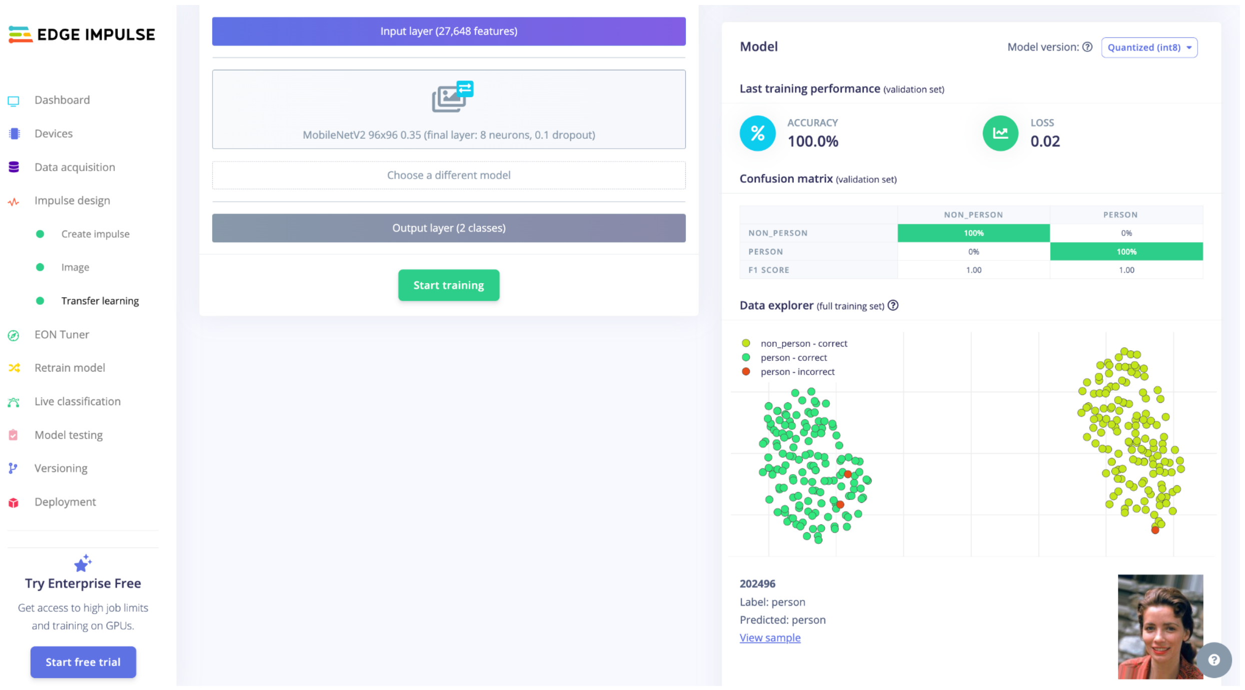
PROCESSING TIME 1 ms.

PEAK RAM USAGE 4 KB

The final stage is **Transfer Learning**.

- Click *Transfer Learning* from the menu bar.
- Select MobileNetV2 96x96 0.35 model and set your parameters. The default parameters work well for our use case but you are welcome to experiment with them (number of training cycles, learning rate, number of neurons) and change them as you deem appropriate. **Please note any changes to these parameters in your writeup.**
- Click *Start Training*. The training can take several minutes depending on the amount of data and the complexity of the features.

When the training process is completed, Edge Impulse will provide statistics about how effective the model is and will show clusters of the training results in the Data Explorer view. You can hover over each point in the cluster and click on the point to see which image generated what result.



MobileNetV2 96x96 0.35 (final layer: 8 neurons, 0.1 dropout)

Last training performance (validation set)

ACCURACY 100.0%

LOSS 0.02

Confusion matrix (validation set)

	NON_PERSON	PERSON
NON_PERSON	100%	0%
PERSON	0%	100%
F1 SCORE	1.00	1.00

Data explorer (full training set)

● non_person - correct
● person - correct
● person - incorrect

202496
Label: person
Predicted: person
View sample

Step 4: Assessing model performance
The final step in the model creation process is to test the model with a labeled test data set to see how well it performs. In this assignment, we will assess each students model with a hidden test dataset. The goal of this assignment is for you to creatively engineer a training dataset for the **MobileNetV2 96x96 0.35 model** to perform person detection in an image. That is, the model should be able to accurately classify whether an image contains a person or not. The test data will contain 96x96 images collected with an Arduino Nicla but with creative modifications (this assignment is supposed to encourage you to think about all the ways that an image can be slightly altered but still maintain the important information that a person is present or not). Additionally, we want you to think about the quality and diversity of the data that you collect. (Hint: It may not be a good idea for your training dataset to only contain images of your face 🙂)

You can create your own testing dataset to assess your model's performance, but we will use the training dataset that you submit to train the **MobileNetV2 96x96 0.35 model** and test it with our hidden dataset.

To assess the model's performance on your test dataset:
- If you asked for an automatic split of your dataset into training and testing, skip to the next step.

    - Otherwise, upload your testing dataset by navigating back to the *Data Acquisition* using the left-hand navigation panel. There, you can upload data for your testing dataset.
- Select *Model Testing* from the left-hand navigation panel.
- Click *Classify All*.

Another remote job will start and run your test data against the trained model, showing you the results upon completion.

Part 2 Deliverables

Submit a .zip folder on Canvas containing:

1. Two folders containing your training data separated into:
    - Person - This folder will contain all images which should be classified as having a person present.
    - Not_Person - This folder will contain all images which should be classified as ***not*** having a person present.
2. A 2-page .pdf write up of your dataset design decisions. **Record the accuracy** of your validation set in your writeup, along with how many files were in your validation set. Also, in your write-up, **summarize the key steps** you took to build and evaluate your custom dataset. Discuss the iterations that you went through to improve the model's ability to detect the presence of a person. For example, you can touch upon some of the items below:
    - How did you curate the dataset? Did you supplement the images that you captured on the Nicla Vision with images from other sources?
    - How many images per class did you collect? Did increasing or decreasing the number of images per class help to increase the accuracy of your model?
    - What do you think about the quality of the dataset you collected? Did you use augmentation, any kind of filtering, or corruption of your training dataset to test robustness?
    - Did you find any corner cases in your dataset that cause an issue with accuracy? What did you do to mitigate it? Did fixing those help in your final accuracy?
    - Did you find any way to fool the model (false-positives)? What do you think was the cause? What improvements do you think can be made to fix this issue?
    - Was the live view tool in OpenMV or EdgeImpulse useful in framing yourself for inferencing? Did it help you combat lighting issues?

The training data submitted for Part 2 of this assignment will be used to train a MobileNetV2 96x96 0.35 model and will be assessed with our hidden test dataset. After grading the assignment, we will announce the top 3 scorers.