CS249r Fall 2023 Assignment 2: Keyword Spotting on Arduino Nicla Vision + Data Preprocessing and Model Optimizations

Assignment Due: Monday, Oct 23, 2023 at 11:59pm on Canvas

Assignment Overview

The purpose of this assignment is to develop an end-to-end keyword spotting (KWS) model using Edge Impulse and deploy it on the Arduino Nicla Vision. You will gain experience with data preprocessing, model optimizations, and embedded deployment.

This assignment is split into two parts:

- In **Part 1**:
  - **What**: You will create an Edge Impulse project, upload the Google Speech Commands (GSC) dataset, extract audio features, train an initial model, and deploy it to the Nicla Vision.
  - **Why**:
    - This is to gain hands-on experience with the Edge Impulse workflow and **understand how data is preprocessed for keyword spotting**.
    - This will also expose you to a **critical dataset** that is widely used in the tinyML community for keyword spotting.
  - **How**: You will use the Edge Impulse platform to extract MFCC features from audio files and train a classification model for keyword spotting. You will then use the Arduino IDE to deploy your keyword spotting model onto the Nicla Vision.
- In **Part 2**:
  - **What**: You will **customize your data preprocessing** and **initial model architecture**. You may also include optimizations like pruning and quantization for efficient deployment.
  - **Why**: The goal is to **understand the tradeoffs** between size, accuracy and latency while varying model architectures, data preprocessing techniques, and optimizations.
  - **How**: You have the choice of **modifying the data preprocessing** and using Edge Impulse's EON tuner to **refine your model architecture and input features**. Another option is to edit the underlying TF Keras code in Edge Impulse to implement custom model optimizations.

New terminologies: Through this exercise you will come across some new concepts and part of the exercise is to help you learn these concepts:

- Mel Frequency Cepstral Coefficients (MFCC)
    - Edge Impulse Tutorial
    - Example Colab
- Spectrograms
    - This video is great
- Confusion matrix

Resources: There are many available resources to help you complete this assignment:

- Responding to your voice (Edge Impulse tutorial)
- Tutorial from Jenny Plunkett (Edge Impulse)
- Slides from Brian Plancher (Professor at Barnard, former TF)
    a. KWS Pre-Processing
    b. KWS Feature-Engineering

Please ask questions (and if you can, answer your fellow classmates' questions) in #assignment2 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 the **Arduino IDE** and capable of detecting keywords from the Google Speech Commands Dataset
ii) a screenshot of the confusion matrix after training your model on Edge Impulse, similar to this one (yours will have different classes):

Part 2: Submit a single zip file containing

i) A one-page writeup of your findings from customizing your data preprocessing/model architecture or model optimizations. Please include any images of the relevant processing blocks on Edge Impulse.

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 must independently optimize your keyword spotting model and complete each of the above deliverables individually.

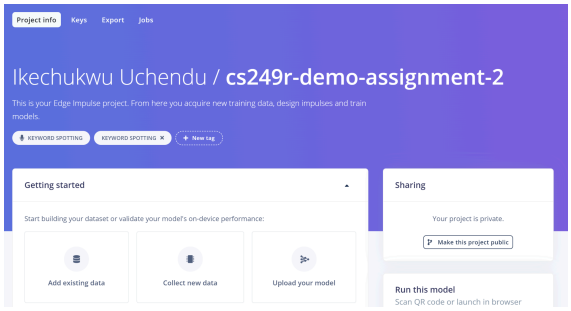
Instructions for Part 1

Uploading the dataset

1. Create a project in Edge Impulse
2. We have prepared a version of the Google Speech Commands V2 (GSC) dataset that is split into train/test/validation
    a. Please download it here
3. **There are ~35 classes in the full dataset. To make model deployment manageable, use the provided `create_datasets.sh` script to extract one keyword of your choosing and label the remaining keywords as "other".**
    a. **For example, to choose "stop" as the keyword you want to recognize, run:**

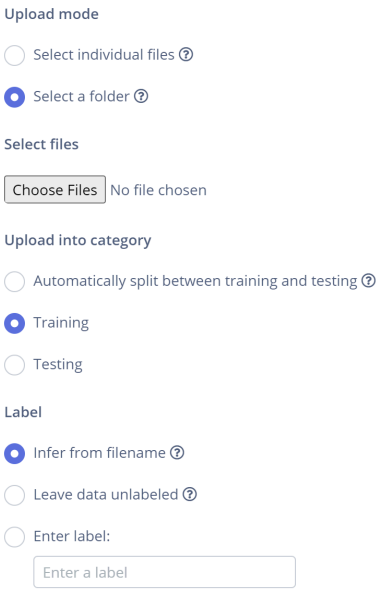
```
chmod +x create_datasets.sh
bash create_datasets.sh stop <num_cores>
```

        i. **Feel free to adjust <num_cores> to speed up moving files depending on your system specifications**
    b. **This will split the data contained in data/train and data/test into two folders - "stop" and "other".**
4. Navigate to your Edge Impulse project
5. Click on "Add Existing Data"



6. Since we have already split the data, we can directly upload the respective train and test directories to edge impulse.

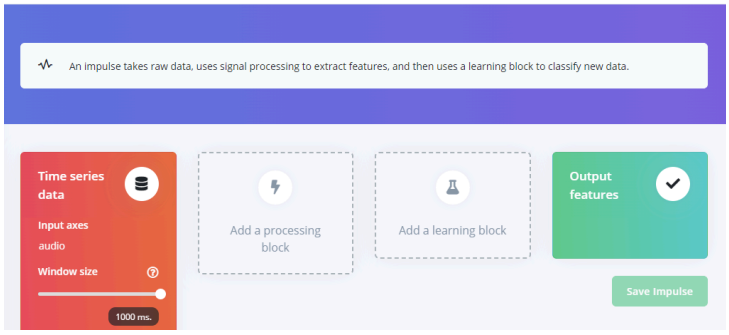
a. **NOTE: Be sure to check the option "Infer from filename"**



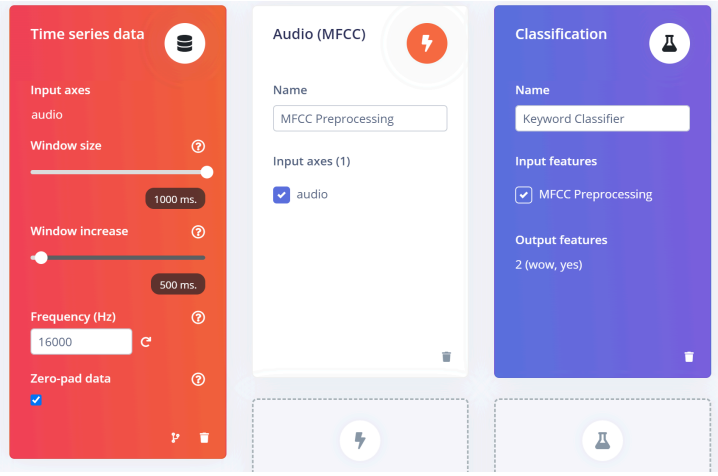
b. **NOTE: The combined train/test datasets are ~500mb, so this upload may take a while**

Designing your impulse

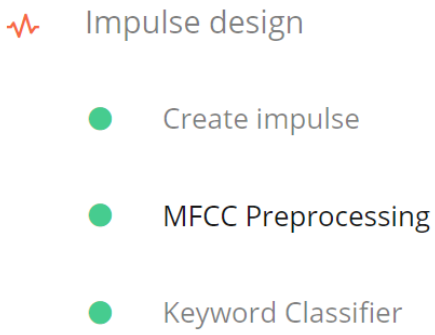
1. Now that your data is uploaded, you can click on "Create Impulse" on the left-hand side of the screen



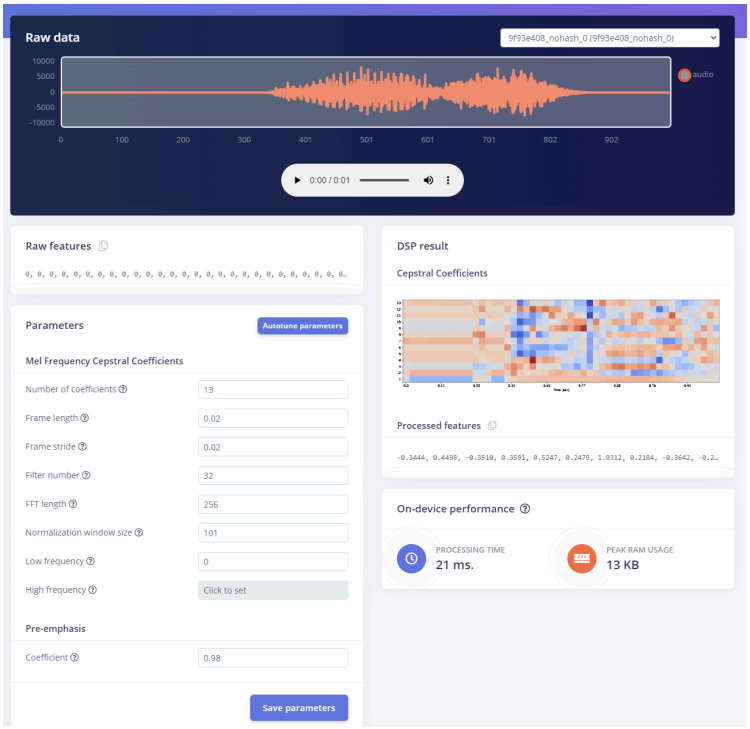
2. Leave the “Time series data” box as default, and add the Audio MFCC processing block, along with the Classification learning block



3. After clicking “Save Impulse”, the options on the left-hand sidebar will change slightly (the options could be different for you depending on how you named your blocks)



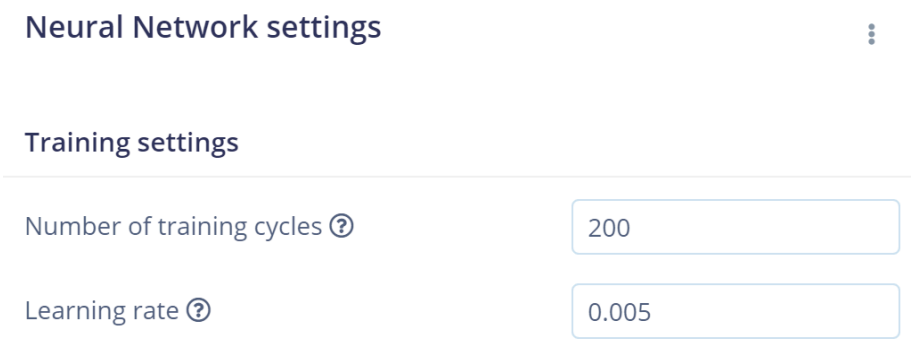
4. Navigate to MFCC Preprocessing
    c. Here you'll be able to listen to various keywords and see their image representation after pre-processing



    d. **Keep all of the parameters default** for now. You will be able to revisit this section in Part 2.
    e. Click "Save Parameters"

Training your keyword spotting model

1. Navigate to Keyword Classifier
    a. Adjust the number of training cycles (epochs) to 200. Feel free to increase the number of cycles to further train your model if desired.



    b. You will see that Edge Impulse provides a nice graphical interface for specifying your model architecture. We'll **leave this as default** for now, but you will be able to edit the architecture in Part 2.
    c. Click "Start training" to begin training your keyword spotting model. After training is done, you should see
2. While your model is training, you can install the Arduino IDE. This will be useful for deploying your keyword spotting model onto the Nicla
3. After the model is trained, you should see a confusion matrix similar to this (yours will have many more classes and data points)

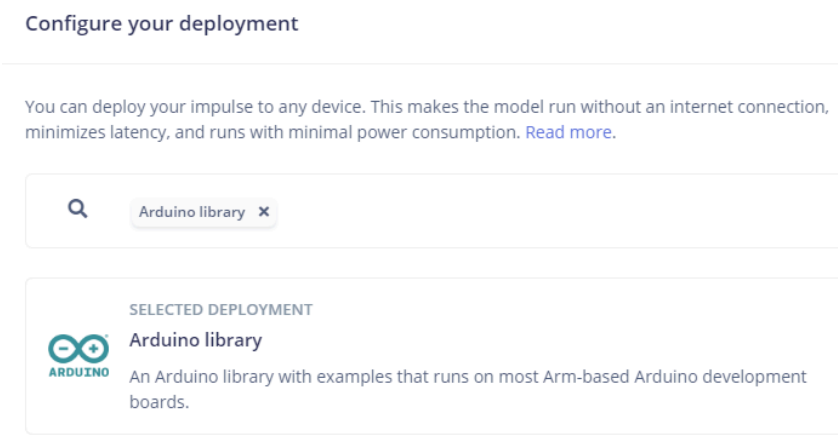
Configuring the Nicla Vision for Audio Input

- Follow these instructions on Edge Impulse to install the firmware your Nicla Vision needs to record audio
    - Complete the instructions for the microphone using the ingestion file `nicla_vision_ingestion_mic.ino`
    - Ignore the instructions for IMU/proximity sensors for now.

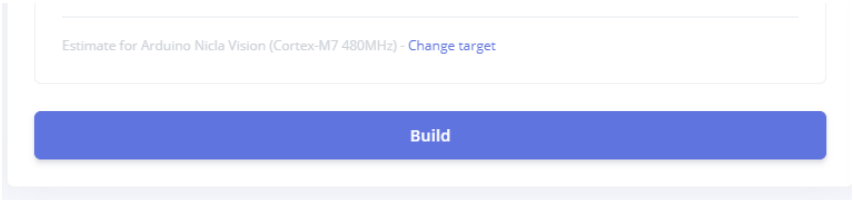
Building the Arduino Library

1. Install the Arduino IDE
2. Click on the “Deployment” tab in Edge Impulse

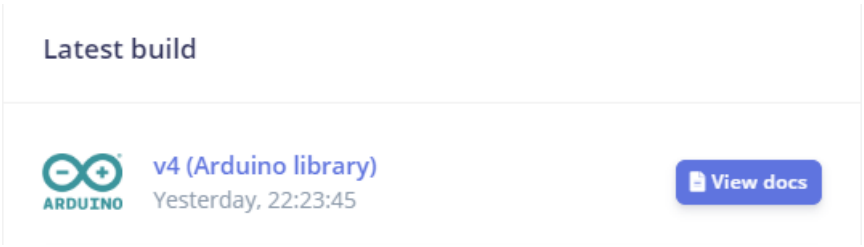
3. Search for “Arduino library”



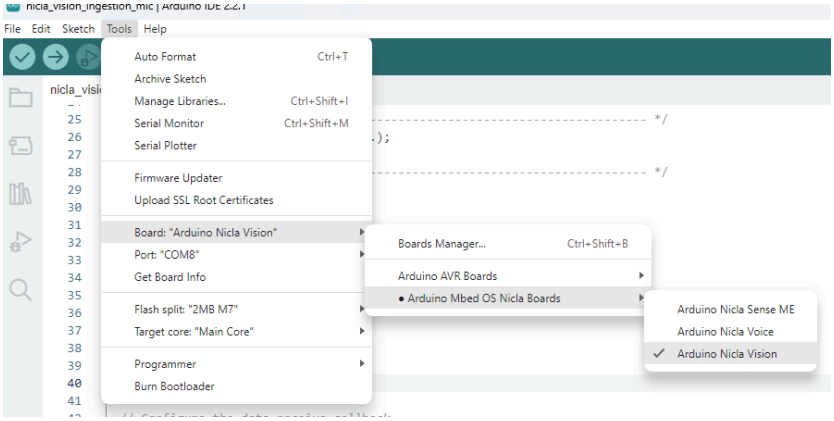
4. Feel free to select the quantized or unoptimized version of your model. Note how the Latency, RAM, and flash change when you go from int8 to float32 and vice-a-versa.
5. Run “compute test accuracy” to determine the difference between the accuracy of the quantized model and the unoptimized model
6. Finally, click “Build”



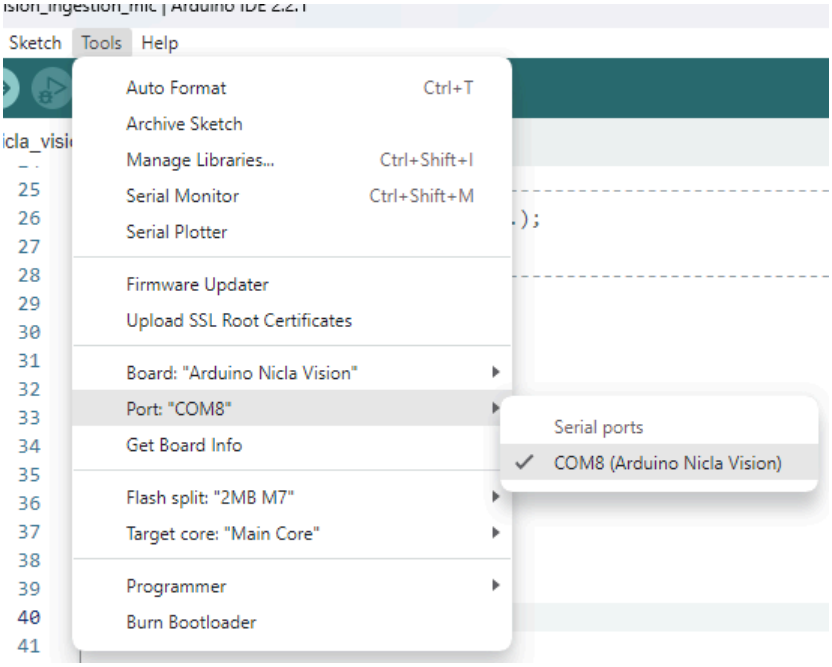
7. After the build is completed, you will be able to download it as a ZIP file:



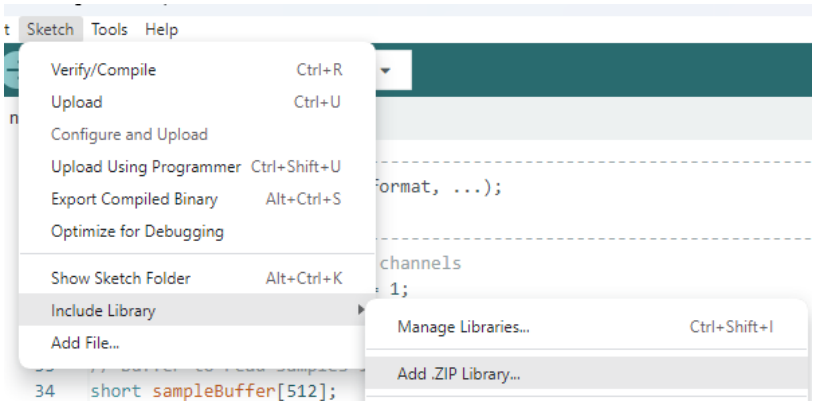
8. Open up the Arduino IDE
9. Make sure to connect the IDE to your Nicla Vision
    a. Ensure that the Board is set to Nicla Vision



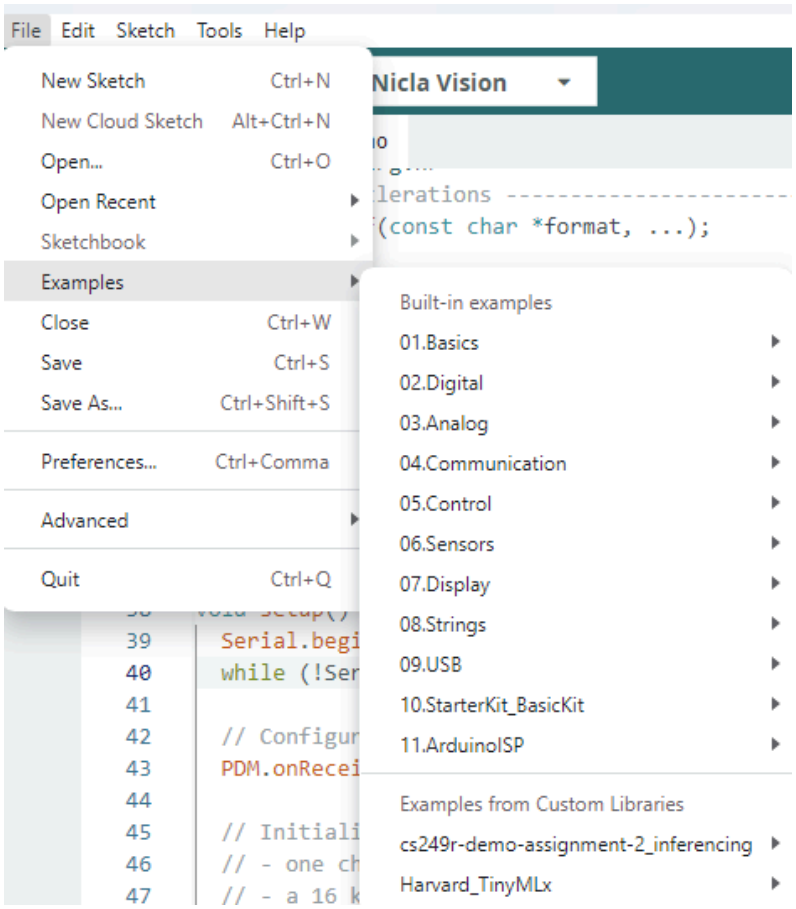
    b. Specify the serial port of your Nicla



10. To add the library with your keyword spotting model, go to Sketch → Include Library → Add .ZIP library

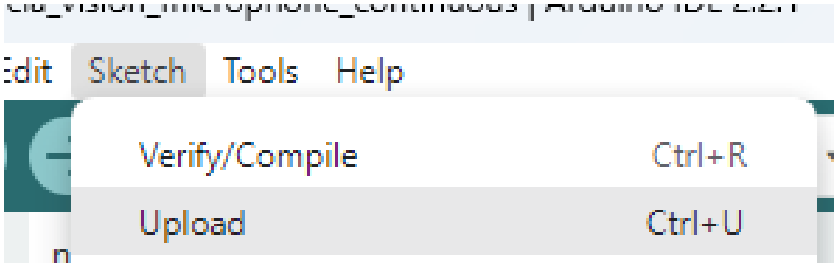


11. Once it is loaded, you can see the library in File → Examples → Examples from Custom Libraries



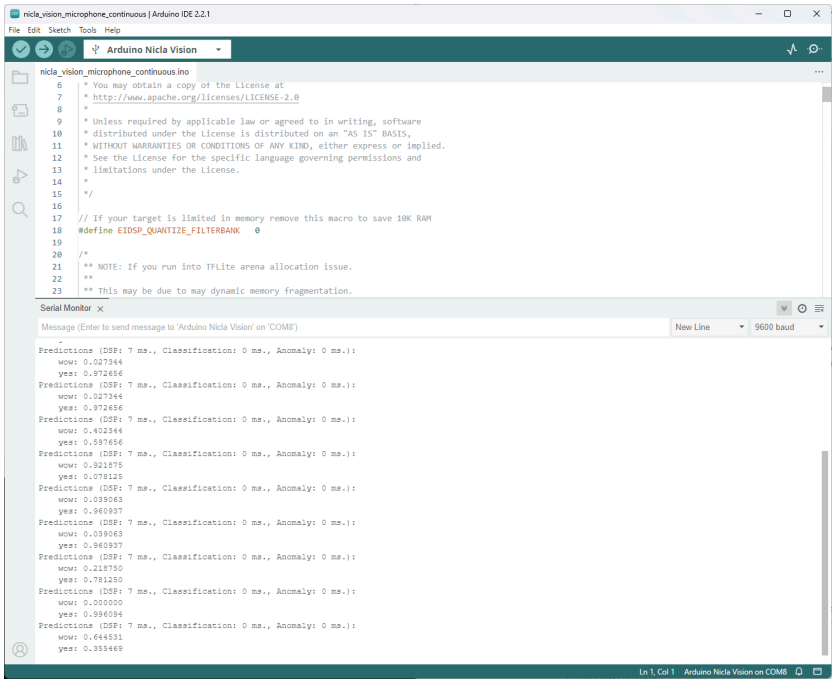
12. Click on **nicla_vision_microphone_continuous** to open up the sketch for continuously running the keyword spotting model

13. Click Sketch → Upload to compile and flash your keyword spotting model to the Nicla



    a. This step took >10 minutes the first time I ran it, so don't worry if it's a bit sluggish

14. Once the upload is complete, open up the Serial Monitor to see the output of your model. Try saying different keywords and see how the predictions change:



Part 1 Deliverables

Submit a .zip file on canvas containing

1. **An .mp4 video of length at most 2 mins showing your Nicla** connected to the **Arduino IDE** and capable of detecting keywords from the Google Speech Commands Dataset
2. A screenshot of the confusion matrix after training your model on Edge Impulse, similar to this one (yours will have different classes):

Instructions for Part 2

In this part, you will customize your model for efficient embedded deployment. To receive full credit, you only need to complete either the Data Preprocessing and Model Architecture modifications **or** the Model Optimization section. You are welcome to complete both options if desired.

Option 1

Data Preprocessing

- Revisit the MFCC Preprocessing block in Edge Impulse and tweak the various parameters such as frame length, frame stride, number of filters, etc.
- Try a different signal processing block such as Spectrograms

Model Architecture

- Use Edge Impulse's EON Tuner to search for viable architectures for your keyword spotting workload
    - Configure the EON tuner to target the Arduino Nicla Vision.
    - Launch an architecture search for your keyword spotting workload
    - In the tuner results page, analyze the various candidate models identified:
        - How do accuracy results compare across different model architectures?
        - Can you identify any patterns in model topology (size, shape, type) that perform better or worse?
    - In your writeup, **include screenshots of the tuner results** and discuss your observations, focusing on high-level trends and tradeoffs between different model architectures.

Option 2

Model Optimization

1. Employ techniques like pruning, quantization, reduced precision, etc. to optimize your keyword spotting model
    a. Tutorials like this one may be helpful

For either option you choose, document the impact on accuracy, latency, and memory usage that your modifications have on the final model.

Part 2 Deliverables

Submit a zip file to Canvas containing:

- A **two-page (max) writeup** that documents your modifications and includes:
    - For each data preprocessing change:
        - Details on what parameter(s) you changed
        - Screenshot of the modified Data Processing block
        - Confusion matrix, accuracy, latency, memory usage
    - For each model architecture change:
        - Details on what layer(s) or parameter(s) you changed
        - Screenshot of the modified Neural Network block
        - Confusion matrix, accuracy, latency, memory usage
    - For each model optimization:
        - Details on the optimization technique used
        - Confusion matrix, accuracy, latency, memory usage