

MonkAI Gesture Detector

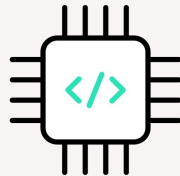
Team Ai Ai Captain
James Jiang and Argie Cunanan

Overview

- 1.) Team Members and Roles
- 2.) Project Goals
- 3.) Workflow
- 4.) Convolutional Neural Network
- 5.) Training the CNN
- 6.) Demo
- 7.) Results

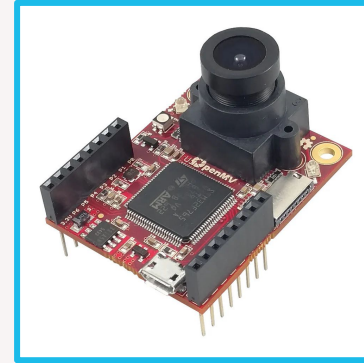
Team Members and Roles

<u>Team Member</u>	<u>Role Title</u>	<u>Role Description</u>
James Jiang	CNN Architecture & Feature Recognition	<ul style="list-style-type: none">- CNN architecture design and feature recognition- Created, trained, and optimized the neural network model using Edge Impulse
Argie Cunanan	OpenMV Integration & Firmware	<ul style="list-style-type: none">- Coded the MicroPython application to load the trained TFLite model, capture images from the OpenMV Cam H7, run on-device inference, and display gesture classification results through serial output

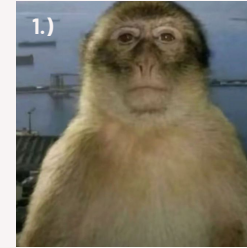


Project Goals

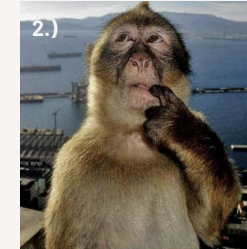
- ❑ Deploy a lightweight ($<100\text{Kb}$), **neural network based** gesture **classifier** on embedded hardware using the **OpenMV H7** microcontroller to recognize hand gestures in real-time.
- ❑ Classify **3** distinct **hand gestures**
- ❑ Run inference on-device (edge) with **minimal latency**
- ❑ Demonstrate with **visual feedback**



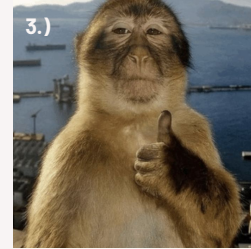
OpenMV H7 Microcontroller



Neutral / no hands



Finger on mouth



Thumbs up

Workflow

Capture: Use OpenMV H7 camera to grab a grayscale frame.



Model: Model: Apply a neural network (exported via Edge Impulse) to classify 1 of 3 gestures.



Output: Print the class over serial and show a matching icon.

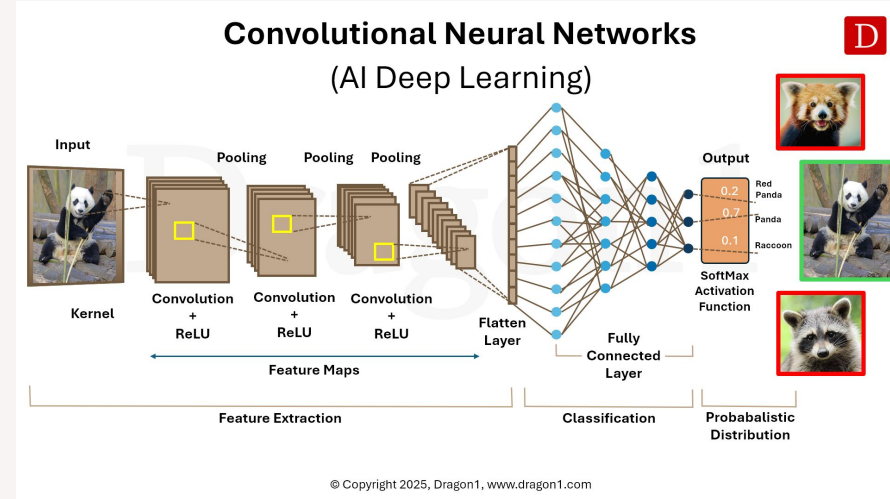


Host demo: Laptop script listens to serial and displays the picture for that gesture.

Convolutional Neural Network

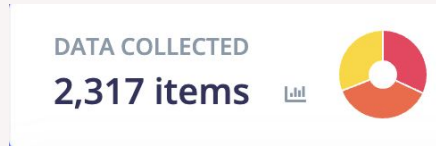
What is a CNN?

- **Convolutional Neural Network (CNN)** is a deep learning model for image recognition
- Uses **convolutional filters** to detect visual features (edges, shapes, textures)
- **Learns** increasingly complex **patterns** through layered feature extraction
- Used in this project to **classify hand gestures** from camera images



Training the Model

- 1.) Captured ~2,000 images of three hand gestures using the OpenMV Cam H7



- 2.) Labeled and uploaded images to Edge Impulse

- 3.) Split data into training (67%) and testing (33%) sets



Training the Model

4.) Created an impulse with image input (96x96) and classification layer

The interface displays four layers in a sequence:

- Image data** (Red layer):
 - Icon: Database symbol
 - Input axes: image
 - Image width: 96
 - Image height: 96
 - Resize mode: Fit short (dropdown)
 - Visual: Two person icons with an arrow between them
 - Trash icon at bottom right
- Image** (White layer):
 - Icon: Lightning bolt
 - Name: Image
 - Input axes (1): image
 - Image: image (dropdown)
 - Trash icon at bottom right
- Classification** (Purple layer):
 - Icon: Erlenmeyer flask
 - Name: Classifier
 - Input features: ☒ Image
 - Output features: 3 (finger_mouth, neutral, thumbs_up)
 - Trash icon at bottom right
- Output features** (Green layer):
 - Icon: Checkmark
 - Output features: 3 (finger_mouth, neutral, thumbs_up)
 - Save Impulse button

Training the Model

5.) Converted images to grayscale to reduce storage and simplify classification

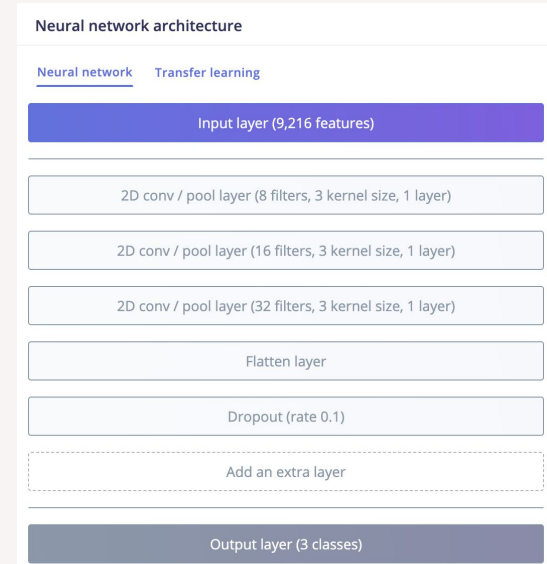
Parameters

Image

Color depth ? Grayscale

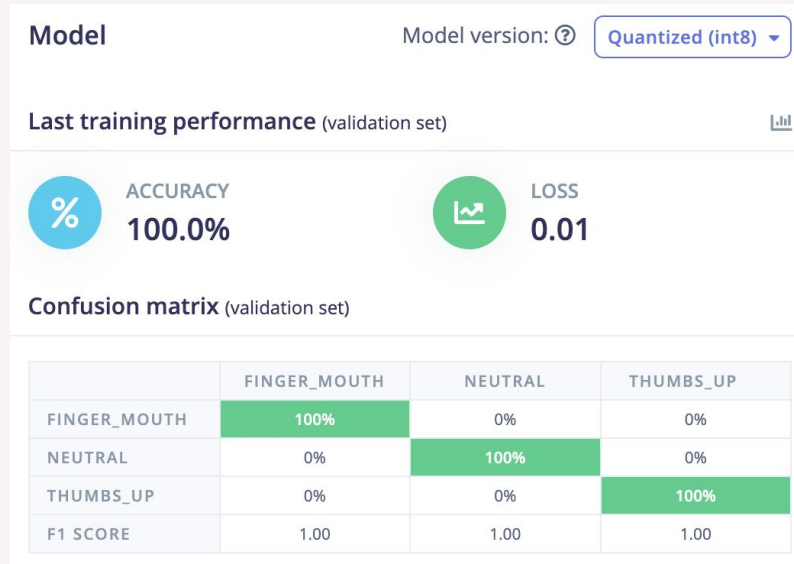
Save parameters

6.) Trained a CNN with three 2D pooling layers for 15 cycles (dropout = 0.1)



Training the Model

7.) Achieved 100% testing accuracy (collected in tandem with training data)



Training the Model

8.) Exported model as TFLite float32

Classifier model

TensorFlow Lite (float32)

80 KB

9.) Deployed model to OpenMV Cam H7

10.) Performed real-time testing and observed classification results



Demo Video



Results



 Inferencing...

thumbs_up (0.96)

Time per inference: 1 ms.



 Inferencing...

finger_mouth (0.85)

Time per inference: 1 ms.



 Inferencing...

neutral (0.92)

Time per inference: 1 ms.

Results



```
6.168968 fps    neutral 0.7045564
6.168972 fps    neutral 0.9310854
6.169012 fps    neutral 0.9598954
6.168612 fps    neutral 0.9634688
6.168652 fps    neutral 0.9279888
6.168694 fps    neutral 0.9688694
6.168698 fps    neutral 0.9730402
6.168702 fps    neutral 0.9615666
6.168742 fps    neutral 0.960658
```



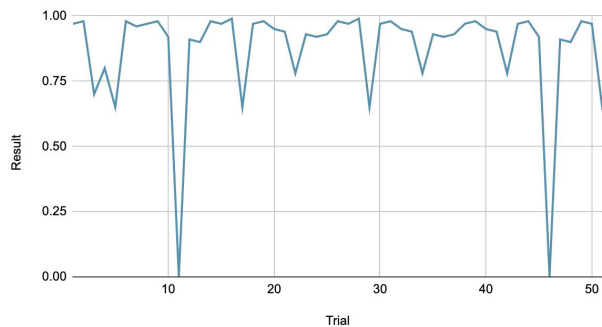
```
6.170214 fps    neutral 0.663359
6.17025 fps     neutral 0.578954
6.16987 fps     neutral 0.660571
6.169908 fps    finger_mouth 0.5898934
6.169946 fps    finger_mouth 0.6129828
6.169948 fps    finger_mouth 0.8005616
6.169948 fps    finger_mouth 0.6694628
6.16995 fps     finger_mouth 0.6694628
6.169988 fps    finger_mouth 0.588427
6.16999 fps     finger_mouth 0.7574586
```



```
6.169362 fps    thumbs_up 0.4635901
6.1694 fps      thumbs_up 0.7790926
6.169404 fps    thumbs_up 0.6235056
6.169442 fps    thumbs_up 0.6214726
6.169446 fps    thumbs_up 0.6574822
6.169484 fps    thumbs_up 0.66232
6.169524 fps    thumbs_up 0.5797426
6.169562 fps    thumbs_up 0.5691718
6.1696 fps      thumbs_up 0.6278942
```

Testing Results

Result vs. Trial



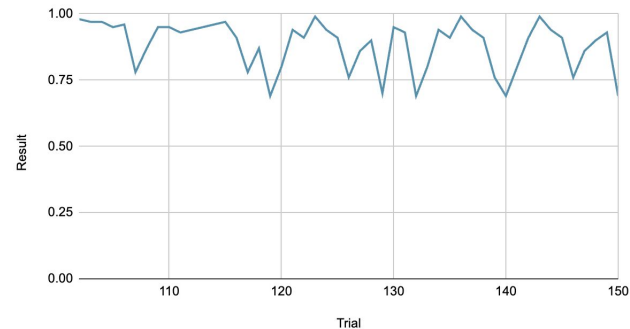
Neutral / No Finger

[AVG] 0.8743137255

Finger on Mouth

[AVG] 0.89466

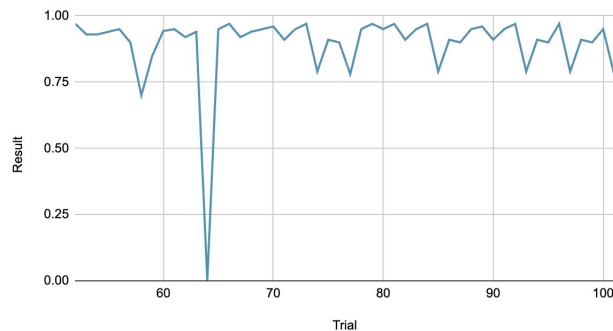
Result vs. Trial



Thumbs Up

[AVG] 0.8834693878

Result vs. Trial







[Overall Average]

$0.884147704433 \times 100\% =$
88.4%

Conclusion

- **Successfully classified** three hand gestures in real time
- Achieved **high accuracy** on the testing dataset
- **Correct gesture detection** on:
 - Edge Impulse local browser testing
 - OpenMV Cam H7 hardware deployment
- **Minimal inference latency** after initial classification
 - Slight delay observed when switching between gestures
- **Grayscale image processing** reduced storage while maintaining accuracy
- Classification **results** displayed **via serial output**

Project Goals

- ✓  Deploy a lightweight, **neural network based** gesture **classifier** on embedded hardware using the **OpenMV H7** microcontroller to recognize hand gestures in real-time.
- ✓  Classify **3** distinct **hand gestures**
- ✓  Run inference on-device (edge) with **minimal latency**
- ❑  Demonstrate with **visual feedback**