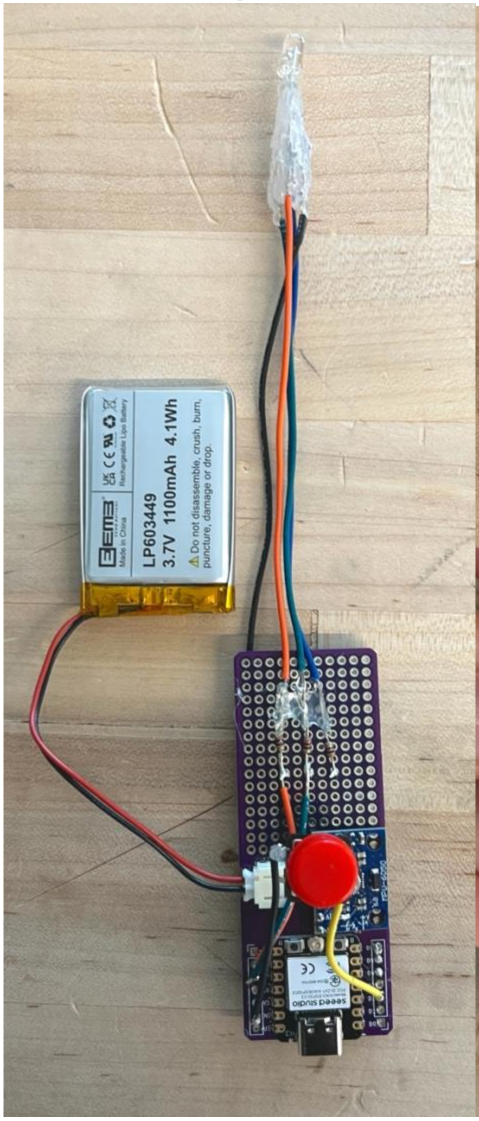
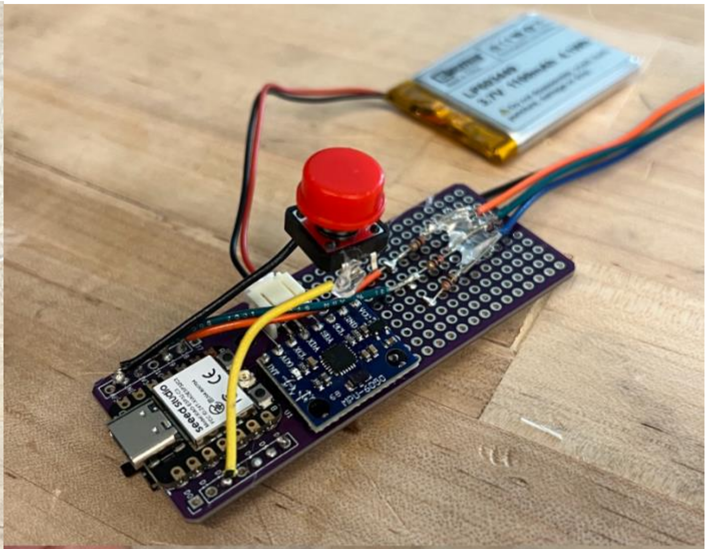
TECHIN 515: Lab 4 Magic Wand Write-Up

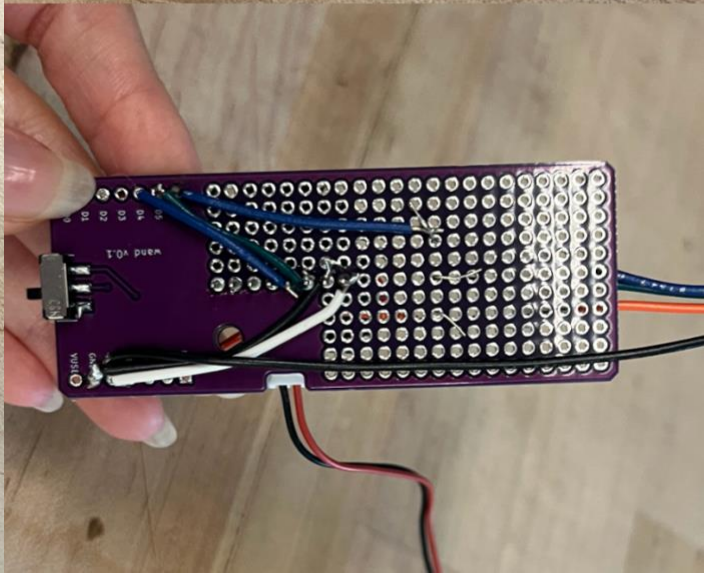
Demo video (enclosure and battery) link:
https://drive.google.com/file/d/1nqKUsfsMG-hMJLx8eQ3tRuU4GqT2saFt/view?usp=sharing (Google Drive)

Demo video of hardware + CLI link:
https://drive.google.com/file/d/1eccm_k3bQcD1eT6j0FWsiqncA27jiykb/view?usp=sharing (Google Drive)

Hardware setup







Part 1: Data Collection

2 Discussion

For this lab, I used my own training data, collecting 100 gestures for each gesture category. While this provided consistent data, using training data collected by multiple students would massively improve the effectiveness, generalization, and reliability. As I was collecting my data, I noticed there was a number of ways to perform the same gesture. Since I naturally gravitated towards a particular style of motion, it would be incredibly difficult for someone else to replicate my gestures exactly, and thus very different results in misclassification. By incorporating data from multiple people, the model is exposed to a wider range of variations, such as the speed of the movement, how big the gesture is, and how they hold the device. This diversity helps make the model more robust to differences in movement patterns – making it more user friendly. This then leads to better usability and reliability.

Part 2: Edge Impulse Model Development

2.1 Design & Implement the Model

Processing block - I chose spectral analysis because it's designed for repeated motion like gestures. This will be able to extract meaningful frequency & power features from the gesture data. It's also lightweight and efficient when compared to other options like raw data and deep learning.

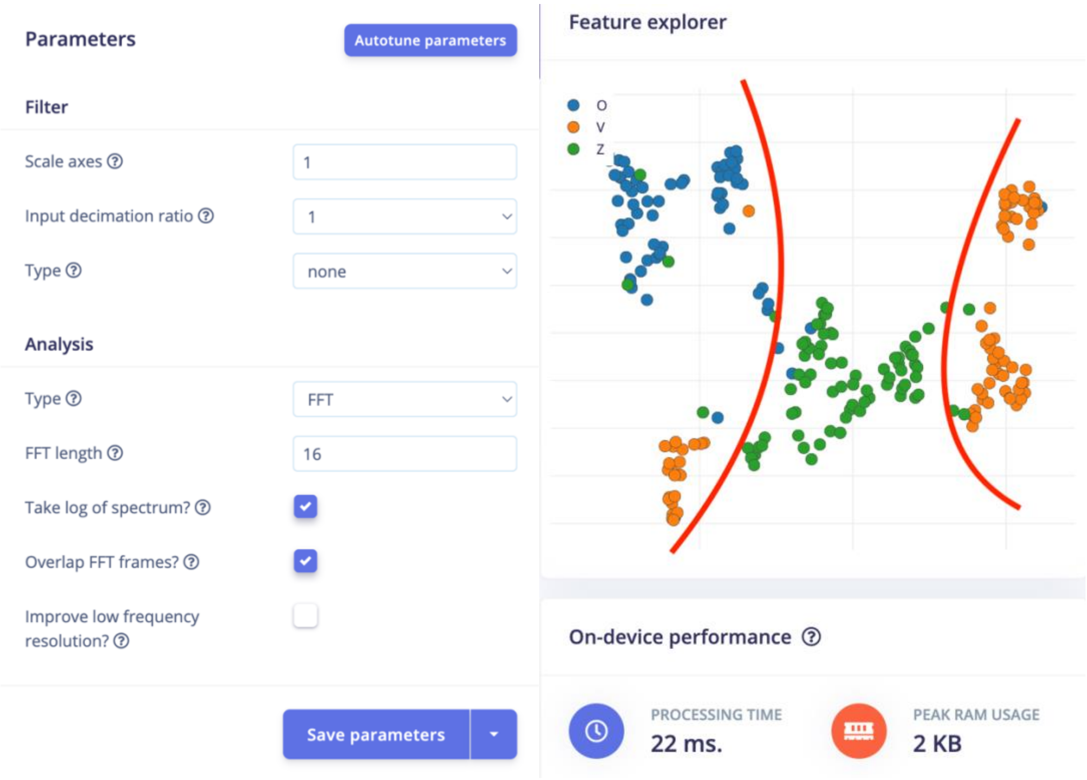
Learning block - I chose classification as this model should be trying to recognise gestures from labelled data. This is important for assigning input data to discrete categories rather than other continuous numerical values.

Window Size Discussion:

- The window size is the data collected and passed through the processing and learning blocks. For this case, this is 1000ms. A larger window size means fewer windows as it covers more time per sample. Whereas, a smaller window means more windows as it covers less time per sample. This results in impact on variation and diversity that the model sees in training.
- The input layer size of the neural network is the number of features produced per window. Larger window sizes means more raw data, translating into more features after preprocessing - this means a larger model that consumes more memory and thus slower inference.
- The effectiveness of slow-changing patterns is that larger windows would be better at capturing slow-changing or complex gestures as they span more time and

capture of the motion. Whereas, shorter windows could miss the full gesture or cut it off midway.

2.2 DSP block



Parameters

Filter

Scale axes: 1

Input decimation ratio: 1

Type: none

Analysis

Type: FFT

FFT length: 16

Take log of spectrum? ☑

Overlap FFT frames? ☑

Improve low frequency resolution? ☐

Feature explorer

On-device performance

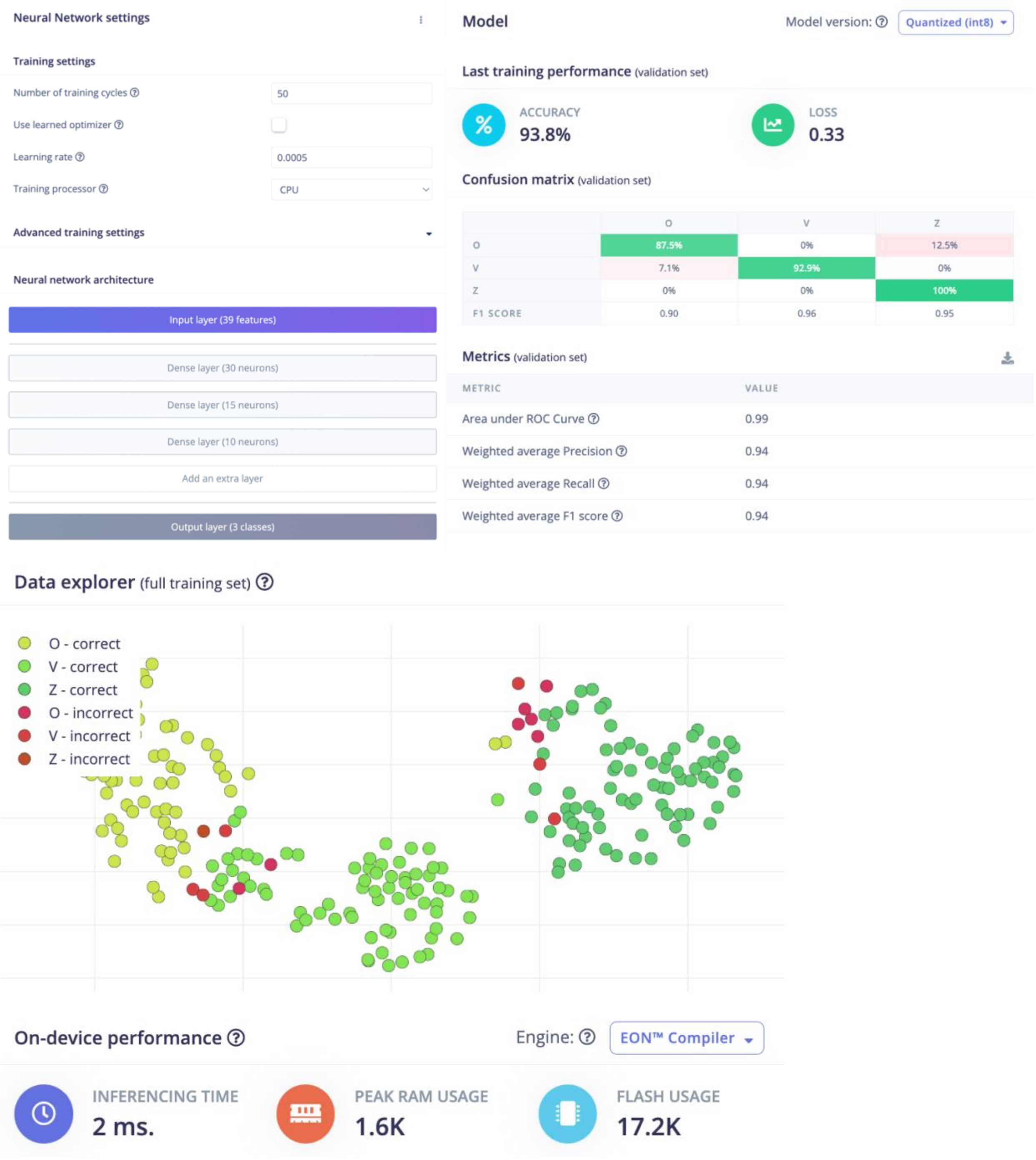
PROCESSING TIME 22 ms.

PEAK RAM USAGE 2 KB

For the DSP block, I chose to use FFT analysis to extract frequency-domain features from the accelerometer data. This is particularly useful for capturing repetitive motion patterns, which often differ more in frequency than in simple time-domain values. The settings I chose were: 1) length 16 for a sufficient frequency resolution, 2) take log of spectrum to enable compression of the FFT output, 3) overlap FFT frames for smoother transitions and to reduce feature loss as gestures are short, 4) no filter type as the gesture signal was already clean and captured under controlled conditions, and 5) 1 for scale axes and input decimation ratio for full resolution across all axes.

With these settings, the results shown on the right demonstrate well-separated clusters for all three gesture classes, indicating that the extracted features are meaningful. Although there seems to be some overlap, the clusters are still dense and mostly consistent, meaning that the model can likely generalize to new samples. For the "V" classifications that lean towards the left where the "O" classifications are above and the "Z" ones to the right, there might have been inconsistencies while collecting the data, or lack of differentiation between V and other gestures, causing the overlap here. Additionally, the performance had a processing time of 22ms and 2KB RAM usage, making it suitable and efficient for real-time deployment.

2.3 ML block



Neural Network settings

Training settings

Number of training cycles ⓘ	50
Use learned optimizer ⓘ	□
Learning rate ⓘ	0.0005
Training processor ⓘ	CPU

Advanced training settings

Neural network architecture

Input layer (39 features)

Dense layer (30 neurons)

Dense layer (15 neurons)

Dense layer (10 neurons)

Add an extra layer

Output layer (3 classes)

Model

Model version: ⓘ Quantized (int8)

Last training performance (validation set)

ACCURACY 93.8%

LOSS 0.33

Confusion matrix (validation set)

	O	V	Z
O	87.5%	0%	12.5%
V	7.1%	92.9%	0%
Z	0%	0%	100%
F1 SCORE	0.90	0.96	0.95

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve ⓘ	0.99
Weighted average Precision ⓘ	0.94
Weighted average Recall ⓘ	0.94
Weighted average F1 score ⓘ	0.94

Data explorer (full training set) ⓘ

On-device performance ⓘ

Engine: ⓘ EON™ Compiler

INFERENCING TIME 2 ms.

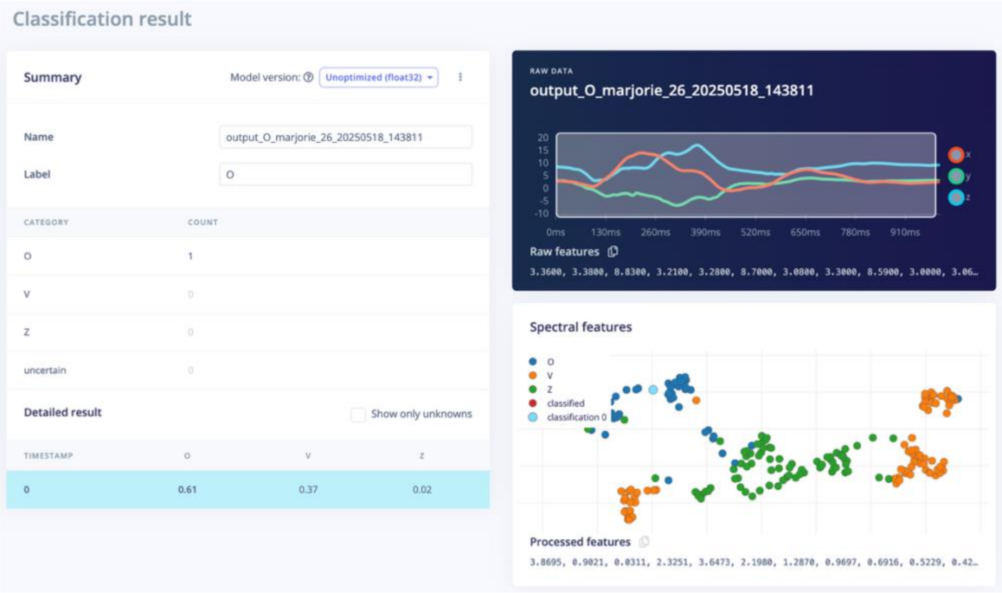
PEAK RAM USAGE 1.6K

FLASH USAGE 17.2K

For the choice of hyperparameters, I first tried the default 30 epochs with 2 dense layers, but this resulted in a 72.9% accuracy with a 0.63 loss – with the main issue being misclassification of class V (14.3% correctly predicted). To improve the accuracy, I increased the learning capacity by increasing the training epochs to 50 and adding to the neural network with an extra dense layer of 10 neurons. This was able to increase the accuracy to 93.8% with a loss of 0.33.

The feature explorer shows well-separated clusters for each gesture class with minimal overlap, suggesting that the features extracted from the FFT block are highly discriminative.

5 Live Classification & Model Testing – Performance Analysis & Metrics



Classification result

Summary

Model version: Unoptimized (float32)

Name: output_O_marjorie_26_20250518_143811

Label: O

CATEGORY	COUNT
O	1
V	0
Z	0
uncertain	0

Detailed result

□ Show only unknowns

TIMESTAMP	O	V	Z
0	0.61	0.37	0.02

RAW DATA

output_O_marjorie_26_20250518_143811

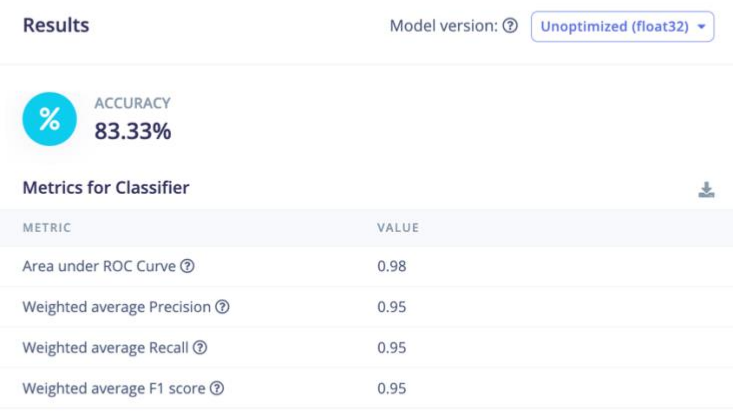
Raw features

```
3.3600, 3.3800, 8.8300, 3.2100, 3.2800, 8.7000, 3.0800, 3.3000, 8.5900, 3.0000, 3.06…
```

Spectral features

Processed features

```
3.8695, 0.9021, 0.0311, 2.3251, 3.6473, 2.1980, 1.2870, 0.9697, 0.6916, 0.5229, 0.42…
```



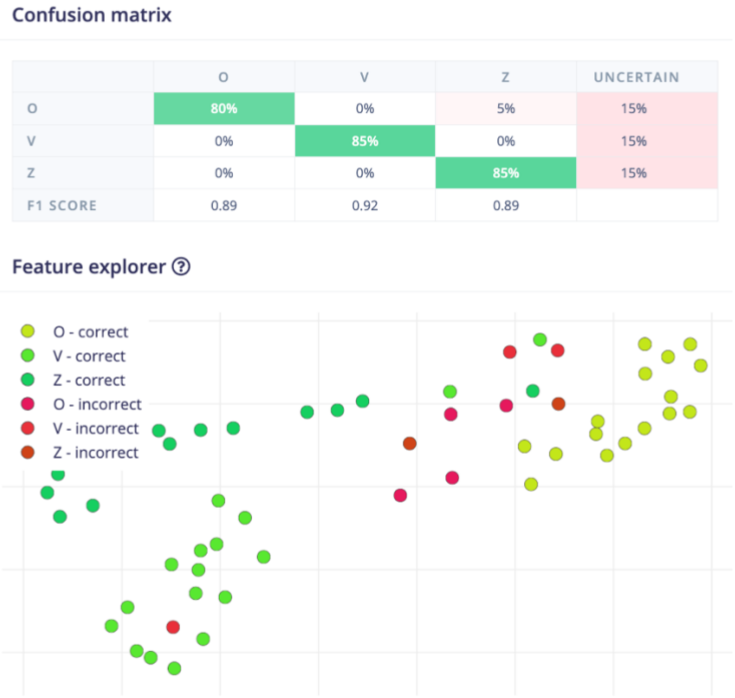
Results

Model version: Unoptimized (float32)

ACCURACY
83.33%

Metrics for Classifier

METRIC	VALUE
Area under ROC Curve	0.98
Weighted average Precision	0.95
Weighted average Recall	0.95
Weighted average F1 score	0.95



Confusion matrix

	O	V	Z	UNCERTAIN
O	80%	0%	5%	15%
V	0%	85%	0%	15%
Z	0%	0%	85%	15%
F1 SCORE	0.89	0.92	0.89	

Feature explorer ⓘ

From the model testing results, the final accuracy is at 83.33%. Key performance metrics included a weighted F1 score of 0.5, precision of 0.95 and recall of 0.95, which indicates a strong balance between minimizing FPs and FNs. Class-wise, the model achieved an F1 score of 0.89 for “Z”, 0.89 for “O” and 0.92 for “V”. The confusion matrix revealed that the most misclassifications were marked as “uncertain” rather than the wrong gesture, suggesting that the model is more cautious. For example, “V” achieved a true positive rate of 85% with uncertain rate of 15% and no false positives. Combined with the low inference time of 2 ms and the 1.6KB of RAM usage from the trained ML results, the model is both relatively accurate and efficient for real-time deployment.

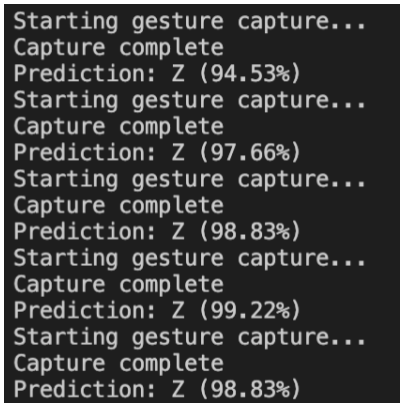
7 Discussion to further enhance performance

Strategy 1: improve quality of training data – collect more samples per class and in these recordings, ensure both diversity and consistency in quality of the data.

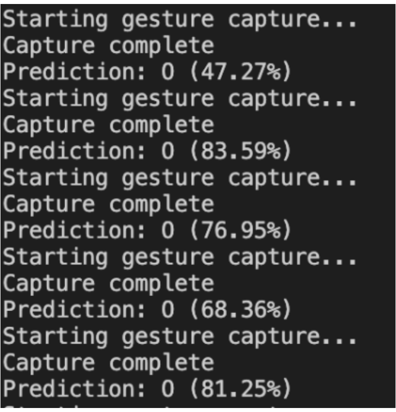
- Strategy 2: Tune model architecture to have more dense layers to increase neurons per layer. Additionally, the epoch could be increased.

Part 3: ESP32 Implementation

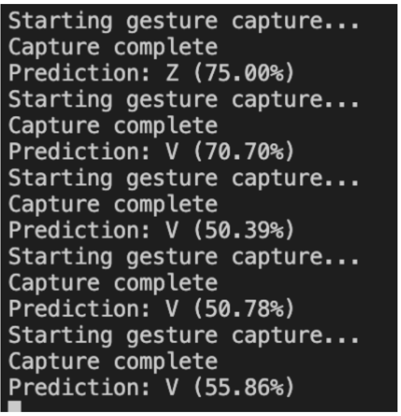
2 Performance Evaluation



```
Starting gesture capture...
Capture complete
Prediction: Z (94.53%)
Starting gesture capture...
Capture complete
Prediction: Z (97.66%)
Starting gesture capture...
Capture complete
Prediction: Z (98.83%)
Starting gesture capture...
Capture complete
Prediction: Z (99.22%)
Starting gesture capture...
Capture complete
Prediction: Z (98.83%)
```



```
Starting gesture capture...
Capture complete
Prediction: O (47.27%)
Starting gesture capture...
Capture complete
Prediction: O (83.59%)
Starting gesture capture...
Capture complete
Prediction: O (76.95%)
Starting gesture capture...
Capture complete
Prediction: O (68.36%)
Starting gesture capture...
Capture complete
Prediction: O (81.25%)
```



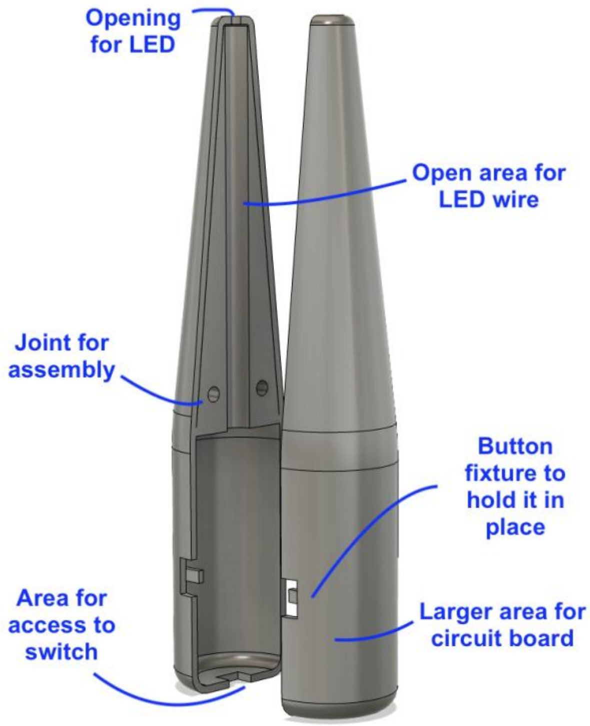
```
Starting gesture capture...
Capture complete
Prediction: Z (75.00%)
Starting gesture capture...
Capture complete
Prediction: V (70.70%)
Starting gesture capture...
Capture complete
Prediction: V (50.39%)
Starting gesture capture...
Capture complete
Prediction: V (50.78%)
Starting gesture capture...
Capture complete
Prediction: V (55.86%)
```

To evaluate the model's real-world performance, I conducted multiple rounds of live classification using the physical wand. For each gesture, I performed 5 repeated trials and recorded the predicted labels and their confidence scores.

- Gesture Z achieved 5/5 in classification, with confidence scores consistently above 94, indicating high confidence and reliability.
- Gesture O achieved 5/5 in classification, with more varied confidence between 47 and 83%, indicating that the model recognizes the gestures well but with lower certainty.
- Gesture V achieved 4/5 in classification, with confidence scores ranging from 50-70%, indicating some misclassification and lower certainty of recognition.

These results show that the model performs very well for Z, well for O and less so for V. Overall, the wand demonstrates strong real-time performance in classification and low latency.

Part 4: Enclosure


Opening for LED
Open area for LED wire
Joint for assembly
Button fixture to hold it in place
Area for access to switch
Larger area for circuit board

Challenges Faced and Solutions

1. Gesture variability – while collecting data, I noticed there were multiple ways to perform the same gesture, and my motion style varied slightly across the samples. To ensure high accuracy, I standardized my gestures during training and tried to ensure consistent speed, orientation and motion range. The orientation was the most important consideration here, as the different angles that I started with mattered a lot. Although this was not a good solution for real-world usability and generalization, for the training process early on, I wanted to reduce noise in a small personal dataset.
2. Gestures for particularly V showed lower confidence or were occasionally misclassified during live classification. To ensure this wouldn't be the case, I tested differences in my gesture, adjusting speed and ensuring more distinct motion paths, which was able to improve the accuracy. Further improvement would involve collecting more diverse data across users so that this wouldn't have to be the solution to this problem.

3. During soldering, some of the exposed wires were getting close to one another, which could have caused serious hardware issues like short circuits – especially during testing within the enclosure. To prevent this from happening, I added hot glue as a quick way to ensure the wires not only stayed in place, but that they wouldn't come into contact, even if there was pressure applied to them.
4. While assembling the enclosure, I struggled with the joint of the two pieces, as the fit was too tight. Because of this, I changed it to be assembled with a rubber band as a quick prototyping fix.