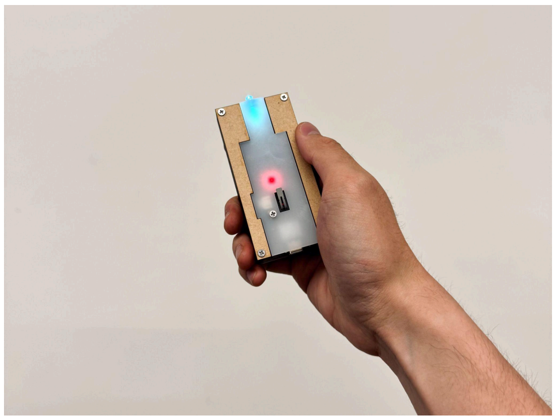
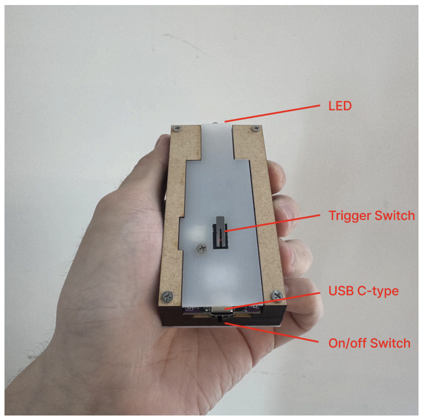
Lab4
Kyle Kim

1. Pictures of hardware setup and connections



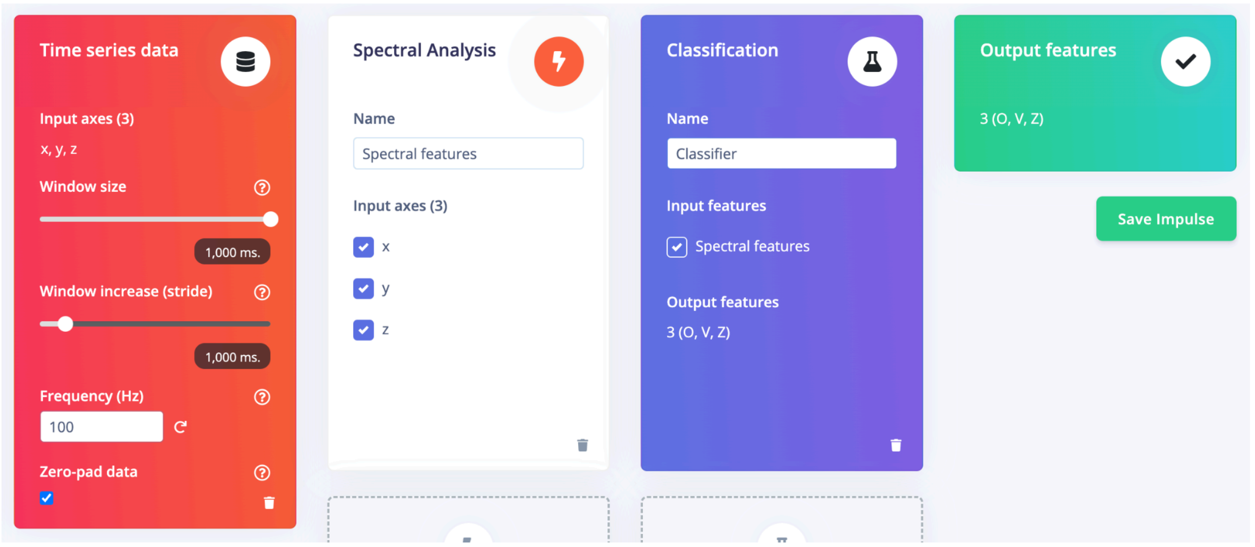


The wand prototype is built using an ESP32 (XIAO C3) board, an MPU6050 IMU sensor, RGB LEDs, and a limit switch as the gesture trigger. The components were connected on a breadboard during development, and then enclosed in a custom MDF and acrylic case. The MPU6050 is connected via I2C: SCL to GPIO22, SDA to GPIO21, with power from 3.3V and GND. The RGB LED is connected to GPIO3, 4, and 5, and the limit switch is connected to GPIO20 with INPUT_PULLUP mode.

2. Data collection process and results

Gesture data was collected using the gesture_capture.ino Arduino sketch and a Python script. Each gesture ("Z", "O", "V") was recorded at 100Hz over a 1-second window, producing 100 samples per instance. At least 20 samples per gesture were collected. While the dataset was collected by a single user, care was taken to introduce variation in gesture direction, size, and speed to simulate diversity and enhance generalization.
The MPU6050 sensor was consistently positioned during all recordings to ensure uniform orientation. Data was saved in CSV format with x, y, z acceleration values.
The resulting dataset was well-separated visually during feature analysis, especially after transforming into spectral features.

3. Edge Impulse model architecture and optimization



Time series data

Input axes (3)

x, y, z

Window size: 1,000 ms.

Window increase (stride): 1,000 ms.

Frequency (Hz): 100

Zero-pad data: ✓

Spectral Analysis

Name: Spectral features

Input axes (3)

✓ x

✓ y

✓ z

Classification

Name: Classifier

Input features

✓ Spectral features

Output features

3 (O, V, Z)

Output features

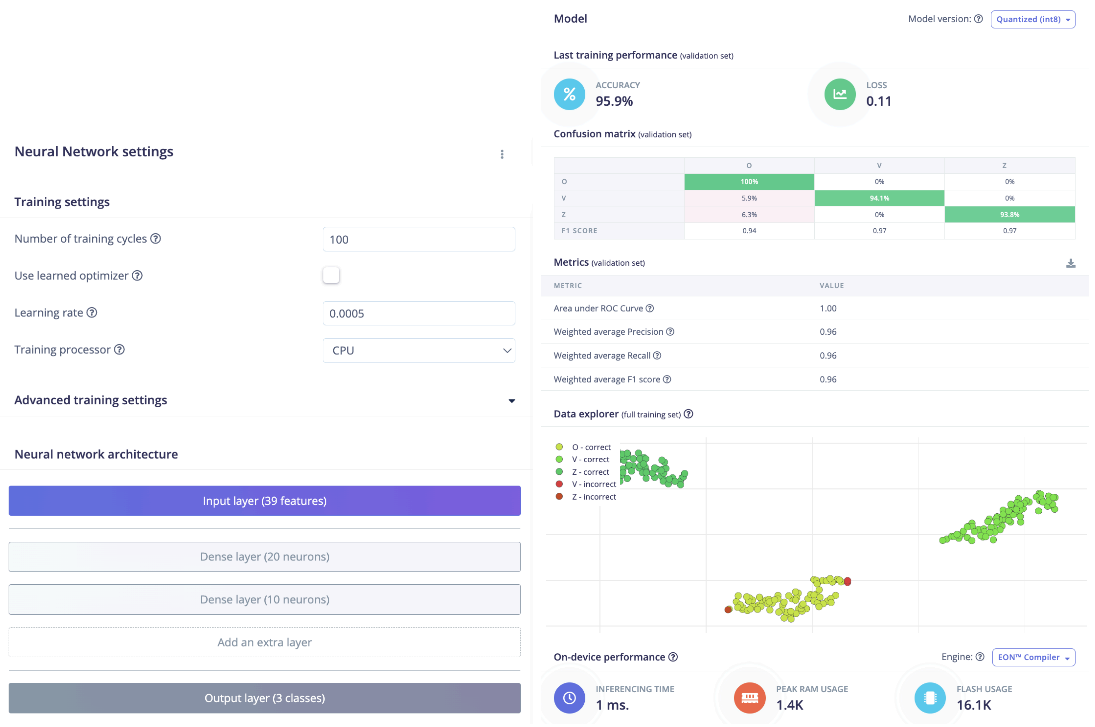
3 (O, V, Z)

Save Impulse

The final impulse was composed of a time-series input (100 Hz) feeding into a Spectral Features DSP block using all three axes (x, y, z). The feature extraction step converted 1-second windows into frequency-domain features, which were then passed into a Classification block. The model outputs one of three predicted gestures: "O", "V", or "Z". The design prioritizes simplicity and efficiency, making it suitable for deployment on ESP32-class microcontrollers.

4. Performance analysis and metrics

- Classifier



Neural Network settings

Training settings

Number of training cycles	100
Use learned optimizer	☐
Learning rate	0.0005
Training processor	CPU

Advanced training settings

Neural network architecture

Input layer (39 features)

Dense layer (20 neurons)

Dense layer (10 neurons)

Add an extra layer

Output layer (3 classes)

Model — Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 95.9%

LOSS 0.11

Confusion matrix (validation set)

	O	V	Z
O	100%	0%	0%
V	5.9%	94.1%	0%
Z	6.3%	0%	93.8%
F1 SCORE	0.94	0.97	0.97

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.96
Weighted average Recall	0.96
Weighted average F1 score	0.96

Data explorer (full training set)

On-device performance — Engine: EON™ Compiler

INFERENCING TIME	PEAK RAM USAGE	FLASH USAGE
1 ms.	1.4K	16.1K

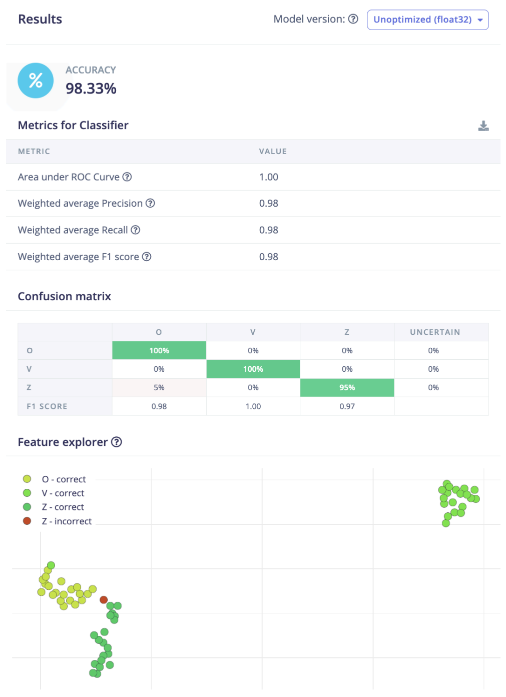
We evaluated the model using both the validation set during training and the final testing set in Edge Impulse.

Classifier Training Results

- Accuracy: 95.9%
- Loss: 0.11
- Weighted Avg Precision / Recall / F1: 0.96
- On-device inference time: 1 ms
- Peak RAM usage: 1.4 KB
- Flash usage: 16.1 KB

Confusion matrix showed near-perfect classification with minor misclassifications between “Z” and “V”.

- Model testing



Results

Model version: Unoptimized (float32)

ACCURACY
98.33%

Metrics for Classifier

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.98
Weighted average Recall	0.98
Weighted average F1 score	0.98

Confusion matrix

	O	V	Z	UNCERTAIN
O	100%	0%	0%	0%
V	0%	100%	0%	0%
Z	5%	0%	95%	0%
F1 SCORE	0.98	1.00	0.97	

Feature explorer

- Accuracy (Test Set): 98.33%
- Weighted Avg Precision / Recall / F1: 0.98

All three gestures achieved very high performance, with perfect recall for “O” and “V”, and 95% for “Z”.

5. Discussions

Q: Why should you use training data collected by multiple students rather than using your own collected data only?
A: Using training data from multiple students increases the diversity and robustness of the model. Each person performs gestures slightly differently due to hand size, motion style, or gesture interpretation. If the model is trained only on one person's data, it may overfit and fail to generalize. Including data from various users ensures the model can recognize gestures across different styles, improving real-world reliability.

Q: Discuss the effect of window size. Consider:

- Number of samples generated
- Number of neurons in input layer
- Effectiveness in capturing slow-changing patterns

A: The window size determines how much time-series data is included per training sample. A larger window can better capture slow-changing or longer gestures, but increases the number of input features and thus the size of the neural network. This may hurt inference speed on low-power devices. A smaller window enables faster inference and smaller models, but may lose important motion context. Choosing the right window size balances feature richness and computational efficiency.

Q: Why did you choose your DSP block?
A: I chose the Spectral Analysis block because it converts raw accelerometer signals into the frequency domain using FFT. This is ideal for gestures, which often have distinct frequency patterns. For example, a sharp zigzag (Z) will differ in frequency content from a smooth circle (O). Spectral features help to distinguish these motion types while keeping feature dimensions compact and efficient.

Q: Why did you choose your learning block?
A: I chose the Classification learning block because it is well-suited for simple, clearly defined tasks like gesture classification with a limited number of classes. This block offers a lightweight architecture optimized for embedded devices like the ESP32. Compared to more complex neural networks, it requires fewer resources and still provides sufficient accuracy for our use case, making it a practical choice for real-time, on-device inference.

Q: Why do you believe the generated features are good enough?
A: After visualizing the 2D feature space, we observed that each gesture class formed relatively distinct clusters, with some overlap. This indicates that the Spectral Features block captured enough variance between gestures. The separation suggests that the model will be able to learn meaningful decision boundaries, supporting good classification performance.

Q: Give at least two potential strategies to further enhance your model performance.
A:

1. Data Augmentation – Introducing small random noise or temporal shifts in the gesture data can make the model more robust to variability during real-world use.\
2. Hyperparameter Optimization – Tuning the number of layers, neurons, learning rate, and activation functions can help the model learn more efficiently and avoid overfitting. Techniques like grid search or cross-validation can be used to find better combinations.

6. Demo video link

https://github.com/kylewskim/TECHIN515_Lab4/blob/main/media/demo.mp4

7. Challenges faced and solutions

When I first tried with Syntiant IMU, I couldn't find 'Arduino Library' on deployment.