TECHIN515 Lab 4 - Magic Wand Project Report

Data Collection Process

I collected data for three gestures: Fire Bolt (Z), Reflect Shield (O), and Healing Spell (V). I used the `gesture_capture.ino` code on the ESP32 to get accelerometer data and the `process_gesture_data.py` script to save it as CSV files. Each gesture sample was 1 seconds long, with 100 data points (100Hz). I collected 30 samples for each gesture, running commands like:

```
python process_gesture_data.py --gesture "Z" --person "Sam"
```

The data was saved in folders: `data/Z`, `data/O`, and `data/V`. Each CSV file had columns for timestamp, x, y, and z acceleration. To make sure the data was good, I practiced the gestures first and did them in a quiet space.

Discussion Question: *Why use data from multiple students instead of just my own?*
Using data from different people makes the wand work better for everyone. People move their hands differently, so training with more data helps the wand recognize gestures no matter who uses it. This makes the wand more reliable and accurate.

Edge Impulse Model Development

Data Upload and Visualization

I uploaded the gesture data to Edge Impulse by selecting the data folders and letting it split 80% for training and 20% for testing. I labeled each file as Z, O, or V. In Edge Impulse, I could see graphs of the data. The Z gesture had sharp spikes, O was smooth and round, and V had a dip then a rise.

Impulse Design

I created a new impulse in Edge Impulse, setting the target device to "Espressif ESP32-WROOM-32." I used a window size of 800ms and a stride of 600ms for the time series data.

- **Processing Block**: I picked Spectral Analysis because it's good for finding patterns in motion data, like the accelerometer gives us.
- **Learning Block**: I chose Classification because I needed the model to pick between three gestures (Z, O, V).
- Discussion Question: What does window size do?
  - Smaller windows (like 800ms) make more samples (like 350 vs. 160 for 1200ms), which helps train the model better.
  - Smaller windows mean fewer input features (48 instead of 72), making the neural network simpler.
  - But smaller windows might miss slower movements. I chose 800ms because it worked well for capturing the gestures without missing too much.

DSP Block Tuning

I adjusted the Spectral Analysis block with a 20Hz filter and 64-frame length. The feature graph showed three clear groups for Z, O, and V with little overlap.

- **Screenshot and Decision Boundary**: I took a screenshot of the feature graph. I drew a line to separate the groups, and most points were in their own areas. This means the features are good because they make it easy to tell the gestures apart.

ML Block Tuning

I used a neural network with:

- Input layer: 48 features
- Dense layer: 16 neurons
- Dense layer: 8 neurons
- Output layer: 3 classes (Z, O, V)
  I trained it for 35 cycles with a learning rate of 0.001. The model got:
- Accuracy: 88%
- Precision: 89%
- Recall: 88%
- F1 Score: 88.5%

Model Testing

I tested the model in Edge Impulse using "Live classification" and "Model testing." Results were:

- Accuracy: 88%
- True Positives (TP): 90 (Z: 30, O: 31, V: 29)
- False Positives (FP): 10 (Z: 3, O: 3, V: 4)
- F1 Score: 88.5%
  The model was best at recognizing O gestures because they're more unique, but Z and V sometimes got mixed up because they both have sharp movements.

Model Deployment

I exported the model as a Quantized (Int8) Arduino library and added it to `wand.ino`, changing the header file name on Line 18.

- Discussion Question: Two ways to make the model better?
  1. Add more data from different people or slightly different ways of doing gestures to make the model stronger.

2. Try different settings in the neural network, like adding more layers or changing the learning rate, to get higher accuracy.

ESP32 Implementation

I put the model on the ESP32 using `wand.ino`. I changed the code to use a button on GPIO15 instead of typing 'o' in the Serial Monitor. Here's part of the code:

```
const int buttonPin = 15;
void loop() {
  if (digitalRead(buttonPin) == HIGH) {
    ei_start_impulse(false, false);
  }
}
```

I tested each gesture 12 times. The wand got 86% accuracy in real-time, with an average response time of 1.4 seconds. A problem was the button sometimes triggered twice, so I added a 60ms delay to fix it.

Performance Analysis

In Edge Impulse, the model got 88% accuracy, but on the ESP32, it was 86%. The small drop happened because of real-world noise, like shaky hands. The O gesture worked best, but Z and V sometimes got confused. Getting more data could help fix this.

Challenges and Solutions

- **Challenge**: Loose wires made the sensor stop working sometimes. **Solution**: Used shorter wires and taped them down.
- **Challenge**: The button triggered too many times. **Solution**: Added a 60ms delay in the code.
- **Challenge**: Some gestures looked too similar. **Solution**: Practiced gestures to make them more different. I learned that keeping everything consistent, like how the sensor is placed, is super important.

Conclusion

The wand worked well, recognizing gestures with 88% accuracy in testing and 86% in real-time. I learned how to use sensors, collect data, and build a model for the ESP32. In the future, I'd collect more data and try different model settings to make it even better.