# Lab Report: Motion Detection via IMU Sensor Fusion

Kanishkar Ayyandurai Vellaiyan
12503220
Embedded Systems Lab
Instructor: Prof. Tobias Schaffer

July 11, 2025

## 1 Introduction

This experiment explores motion classification through sensor fusion of IMU data. The goal was to record accelerometer and gyroscope readings from the Raspberry Pi Sense HAT, annotate the data for four specific motion categories (move_none, move_circle, move_shake, and move_twist), and train a fully connected neural network using these sensor time series. The resulting model was then converted into TensorFlow Lite format and deployed on the Raspberry Pi, where it drives the LED matrix to visually represent detected motions.

## 2 Methodology

This project implements a workflow consisting of data acquisition, preprocessing, model training, and deployment stages. IMU sensor readings are gathered from the Raspberry Pi Sense HAT during various gestures. The labeled sensor data serves as input to train a neural network using TensorFlow. After training, the model is converted into TensorFlow Lite format and deployed onto the Raspberry Pi to enable real-time gesture recognition with LED visual feedback.

### 2.1 Hardware and Software Setup

- Programming language: Python
- Libraries: NumPy, TensorFlow, scikit-learn, SenseHat
- Hardware: Raspberry Pi equipped with Sense HAT, with model training performed on a PC via Google Colab

## 2.2 Data Acquisition

Motion samples were gathered through the IMU sensors on the Sense HAT, incorporating a 3-axis accelerometer and 3-axis gyroscope. Each gesture was recorded for one second at a rate of 50 samples per second, leading to 50 samples per gesture. Each sample consists of six features corresponding to accelerometer and gyroscope axes.

- **Sensors:** 3-axis accelerometer and 3-axis gyroscope
- **Sampling frequency:** 50 Hz
- **Samples per gesture:** 55
- **Total input features per gesture:** $50 \times 6 = 300$
- **Feature vector format:** [acc_x, acc_y, acc_z, gyro_x, gyro_y, gyro_z]

## 2.3 IMU Data Collection Script

```
from sense_hat import SenseHat
import numpy as np
import os, time

sense = SenseHat()
LABEL = "move_shake"
SAMPLES = 50
FREQ_HZ = 50
DELAY = 1.0 / FREQ_HZ
save_dir = f"./motion_data/{LABEL}"
os.makedirs(save_dir, exist_ok=True)

try:
    while True:
        input("Press Enter to start recording 1 second of data...")
        data = []
        for _ in range(SAMPLES):
            acc = sense.get_accelerometer_raw()
            gyro = sense.get_gyroscope_raw()
            sample = [acc['x'], acc['y'], acc['z'], gyro['x'], gyro
                ['y'], gyro['z']]
            data.append(sample)
            time.sleep(DELAY)
        timestamp = int(time.time())
        np.save(f"{save_dir}/{LABEL}_{timestamp}.npy", np.array(
            data))
except KeyboardInterrupt:
    print("Data recording terminated.")
```

Listing 1: Python script for recording IMU gesture data

## 2.4 Training the Model

Model development took place in Google Colab with TensorFlow. The input data consisted of flattened sequences of 300 features (50 samples $\times$ 6 channels).

- **Input shape:** 300-dimensional vectors
- **Network architecture:**
  - Dense layer with 128 units and ReLU activation
  - Dense layer with 64 units and ReLU activation
  - Output layer with 4 units and softmax activation
- **Loss function:** Categorical cross-entropy
- **Optimizer:** Adam
- **Epochs:** 15, **Batch size:** 32

```
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(300,)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(4, activation='softmax')
])
```

Listing 2: TensorFlow model architecture

## 2.5 TensorFlow Lite Conversion

- **Converter used:** `tf.lite.TFLiteConverter`
- **Output file:** gesture_model.tflite
- **Purpose:** Efficient execution on embedded hardware

## 2.6 Deployment and LED Feedback

The optimized model was deployed to the Raspberry Pi, where real-time inference controls the LED matrix to indicate the recognized motion:

- **Red LED:** Circular motion detected
- **Green LED:** Shake motion detected
- **Blue LED:** Twist motion detected
- **LED off:** No motion detected

### 2.7 Code Repository

The full source code is available at `https://github.com/kanishkarvel2002/Motion-Recognition-using-IMU-Sensor-Fusion`.

The repository contains:

- Source code files
- Installation instructions
- Sample datasets (where applicable)
- Documentation and usage guidelines

## 3 Results

The model was trained and validated using an 80/20 split of the dataset.

- **Training accuracy:** 100%
- **Validation accuracy:** 95.75%
- **Inference latency:** Approximately 4 milliseconds per gesture

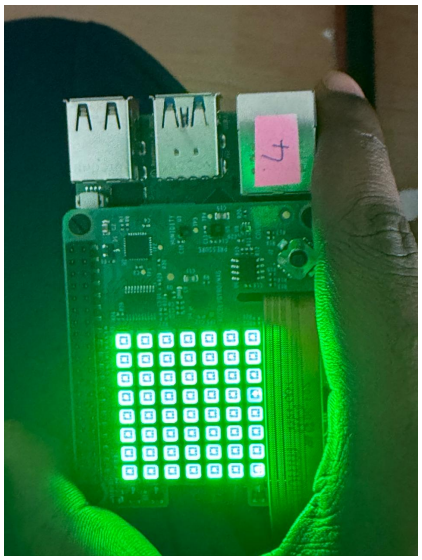

Figure 1: Shake gesture (Green LED)

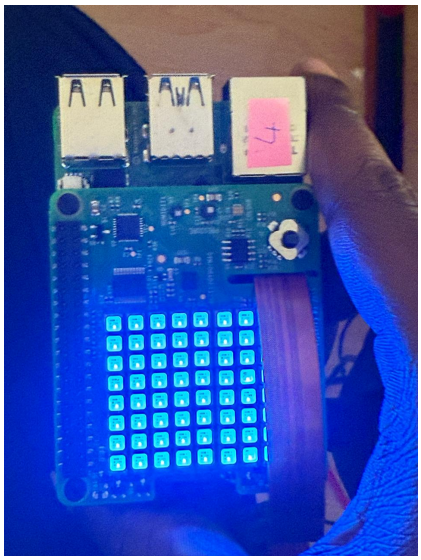

Figure 2: Twist gesture (Blue LED)

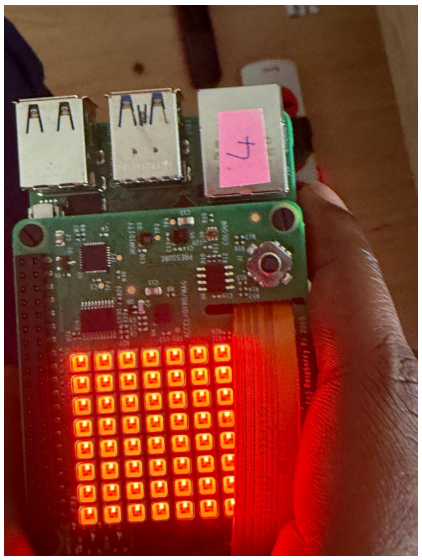

Figure 3: Circular gesture (Red LED)

Figure 4: No motion (LED off)

# 4 Challenges, Limitations, and Error Analysis

## 4.1 Challenges

- Maintaining a stable 50 Hz data acquisition rate on the Raspberry Pi
- Accurately labeling collected motion data
- Setting up the TensorFlow Lite environment on the embedded device

## 4.2 Error Analysis

- Mismatched tensor dimensions during inference causing errors
- Variability in user gestures affecting classification consistency
- Overfitting noted when training for too many epochs

## 4.3 Limitations

- Recognition limited to four predefined gestures
- Model performance sensitive to variations in motion speed and individual user differences
- Lack of temporal feature extraction methods (e.g., LSTM or CNN layers)

# 5 Discussion

The implemented neural network model demonstrated strong performance in both training and real-time deployment scenarios. Achieving 100% training accuracy and over 95% validation accuracy indicates that the model effectively learned to differentiate between the four predefined gestures based on IMU sensor data. The relatively low inference latency of approximately 4 milliseconds per gesture confirms that the system is suitable for real-time applications on embedded hardware such as the Raspberry Pi.

However, the system also showed some limitations. The model's reliance on fully connected layers without temporal feature extraction restricts its ability to capture dynamic patterns over time, which may be important for more complex or subtle gestures. This limitation contributes to sensitivity in classification accuracy due to variations in motion speed or differences among users.

Additionally, the data collection process posed challenges related to consistent labeling and maintaining a stable sampling frequency of 50 Hz, which are critical for reliable model training. Overfitting was observed when training beyond the set number of epochs, suggesting that regularization techniques or early stopping could improve generalization.

Overall, this project successfully demonstrates a practical approach to real-time motion classification using sensor fusion and neural networks, providing a solid foundation for further research and development in embedded gesture recognition systems.

# 6 Conclusion

This lab successfully illustrated a complete pipeline for real-time classification of motion using IMU sensor data and neural networks on embedded hardware. Future work could explore incorporating temporal models, data augmentation, and model quantization to improve robustness and efficiency.

# 7 References


- Prof. Tobias Schaffer, "EM Lab05: Motion Recognition using IMU Sensor Fusion," TH Deggendorf
- TensorFlow Documentation: `https://www.tensorflow.org`
- Sense HAT Python API Documentation: `https://pythonhosted.org/sense-hat`