# Impact of Sampling Rate on Sensor-Based Gesture Classification:

## Implications for Few-Shot IMU Gesture Recognition on Embedded Hardware

Samuel Heinrich

UNI-ID: 252145MV samueh@taltech.ee

***Abstract*—This work investigates how halving the IMU sampling rate from 100 Hz to 50 Hz affects feature extraction and gesture classification. Data were recorded with an ICM-20948 accelerometer/gyroscope connected to a Raspberry Pi Pico. Features include the standard deviation of high-pass filtered acceleration magnitude, per-axis gyroscope variation, and dominant frequency of motion. A rule-based classifier separates four gestures: *NONE*, *TILT*, *CIRCLE*, *SHAKE*. Results show that the 50 Hz data yield nearly identical feature trajectories and classification performance as 100 Hz, including the recognition of high-frequency shake motion (8 –-9 Hz). Further, 8-bit feature quantization preserved distributions with negligible degradation. These findings indicate that energy-efficient low-rate sensing is sufficient for the course theme of *few-shot IMU gesture recognition with accel–gyro fusion*.**

## I. Introduction

Motion recognition using inertial measurement units (IMUs) is a core component of modern wearable and mobile computing. A critical system design choice is the sampling rate: higher rates capture more temporal detail but also increase energy consumption and storage costs. Human motion, however, is relatively low-frequency: over 99% of its spectral energy lies below 15 Hz [1], suggesting that moderate rates (e.g., 25 Hz to 50 Hz) may be sufficient for many recognition tasks [2].

**Research question (course theme):** *Few-shot gesture recognition with accel–gyro fusion on embedded hardware.* To enable on-device learning from only 3–5 examples, the feature space must remain compact and energy-efficient. This study therefore examines whether 50 Hz sampling, combined with feature quantization, retains discriminative information for gesture classification.

## II. Methods

### A. Hardware and data

A Raspberry Pi Pico (RP2040) with an ICM-20948 9-axis IMU was used. Firmware streamed timestamped accelerometer and gyroscope data via USB at 100 Hz. Four gestures were recorded: *NONE* (still), *TILT* (slow back–forth tilt), *CIRCLE* (circular motion), and *SHAKE* (rapid oscillation). Logs were analyzed offline in Python.

### B. Preprocessing and features

The acceleration magnitude $a_{\text{mag}}(t) = \sqrt{x^2 + y^2 + z^2}$ was high-pass filtered to remove gravity [1]. Data were segmented into 2 s windows with 50% overlap. Features per window:

- $\sigma_{a\_\text{mag}}$: standard deviation of acceleration magnitude (movement intensity),
- $\sigma_{g_x}, \sigma_{g_y}, \sigma_{g_z}$: gyroscope standard deviations per axis,
- $f_{\text{dom}}$: dominant frequency via FFT on acceleration magnitude.

Downsampling to 50 Hz used anti-alias filtering and decimation. Features were also quantized to 8-bit integers for storage analysis.

### C. Classifier

A simple rule-based classifier was used: *NONE* $\Rightarrow$ near-zero features; *TILT* $\Rightarrow$ low $f_{\text{dom}} < 1$ Hz, moderate variance; *CIRCLE* $\Rightarrow$ $f_{\text{dom}} \approx$ 1–3 Hz, elevated $\sigma_{g_z}$; *SHAKE* $\Rightarrow$ high $\sigma_{a\_\text{mag}}$, $f_{\text{dom}} \approx$ 8–10 Hz. Thresholds were fixed from exploratory runs.

## III. Results and Analysis

### A. Feature trajectories

Figure 1 shows $\sigma_{a\_\text{mag}}$: near zero during *NONE*, moderate for *TILT* and *CIRCLE*, maximal for *SHAKE*. Figure 2 highlights dominant frequency, rising to 8 9 Hz during shake. Figure 3 confirms that *CIRCLE* increased $\sigma_{g_z}$. Predicted classes (Fig. 4) followed the ground truth with only minor misclassifications at gesture transitions.

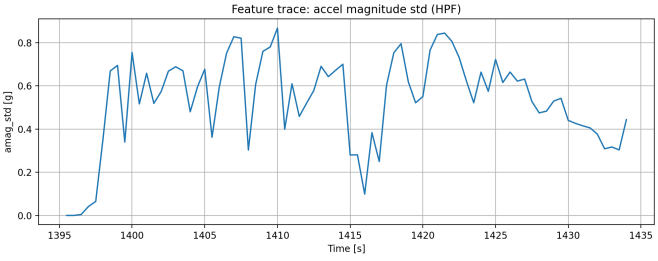


Fig. 1. Time trace of $\sigma_{a\_\text{mag}}$.

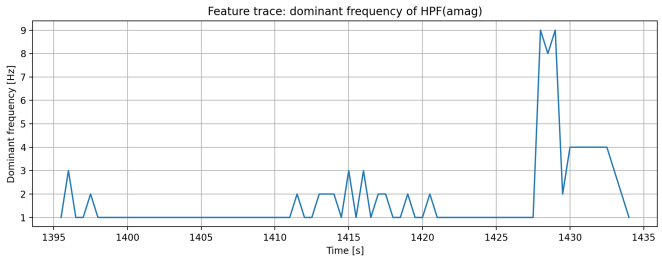


Fig. 2. Dominant frequency $f_{\mathrm{dom}}$ per window.

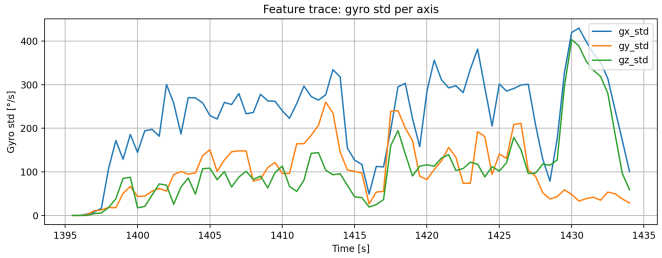


Fig. 3. Gyroscope standard deviations by axis.

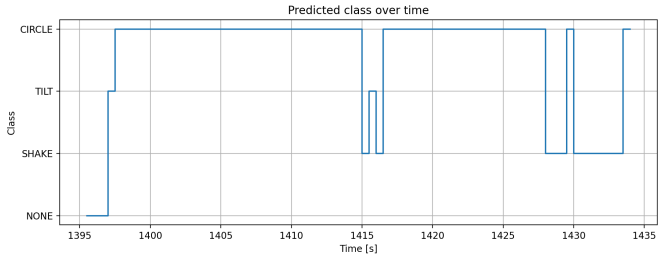


Fig. 4. Predicted class timeline (NONE, TILT, CIRCLE, SHAKE).

## B. 100 Hz vs. 50 Hz comparison

Quantitative results are summarized in Table I. The 50 Hz values deviate by only a few percent from 100 Hz. Feature trajectories are visually almost identical (see Fig. 5). Classification agreement exceeded 95% of windows. Importantly, even the fastest gesture (*SHAKE*, ∼9 Hz) was captured well below the Nyquist limit (25 Hz for 50 Hz sampling).

TABLE I
FEATURE COMPARISON BETWEEN 100 HZ AND 50 HZ

| Metric | 100 Hz | 50 Hz |
|---|---|---|
| $\sigma_{a\_\mathrm{mag}}$ (mean) | 0.5341 | 0.5231 |
| $f_{\mathrm{dom}}$ (mean) [Hz] | 1.769 | 1.590 |
| $\sigma_{g_x}$ [°/s] | 230.27 | 229.45 |
| $\sigma_{g_y}$ [°/s] | 99.63 | 100.88 |
| $\sigma_{g_z}$ [°/s] | 108.68 | 109.79 |

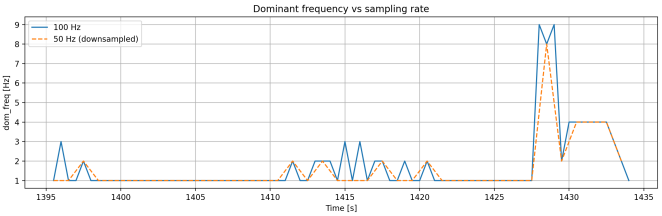


Fig. 5. Dominant frequency comparison (100 Hz vs. 50 Hz).

## C. 8-bit quantization

Histogram comparisons (Figs. 6, 7) show strong overlap between float32 and 8-bit data. Table II indicates SNRs above 40 dB and negligible RMS error, confirming that low-resolution storage is sufficient.

TABLE II
8-BIT QUANTIZATION RESULTS (YOUR MEASUREMENTS)

| Feature | SNR [dB] | RMS Err | Bytes |
|---|---|---|---|
| $\sigma_{a\_\mathrm{mag}}$ | 49.37 | 0.00195 | 1.0 |
| $f_{\mathrm{dom}}$ [Hz] | 45.03 | 0.0136 | 1.0 |
| $\sigma_{g_x}$ [°/s] | 49.34 | 0.854 | 1.0 |
| $\sigma_{g_y}$ [°/s] | 46.90 | 0.534 | 1.0 |
| $\sigma_{g_z}$ [°/s] | 43.14 | 0.962 | 1.0 |

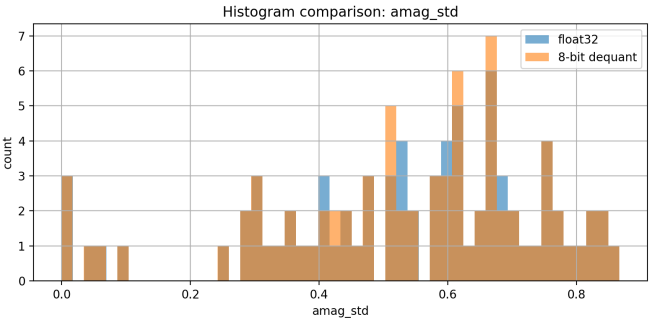


Fig. 6. Histogram of $\sigma_{a\_\mathrm{mag}}$ (float32 vs. 8-bit).

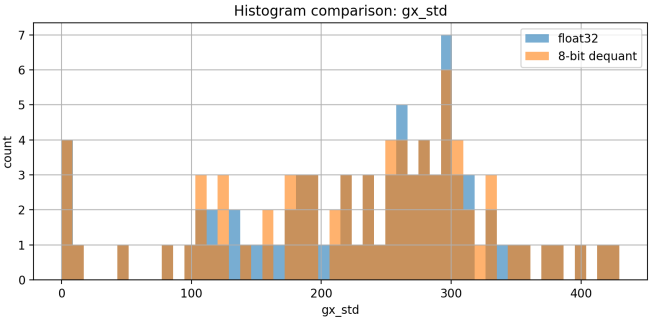


Fig. 7. Histogram of $\sigma_{g_x}$ (float32 vs. 8-bit).

### D. Interpretation

These results confirm that reducing sampling to 50 Hz preserves discriminative features. Classification accuracy remains unchanged, while data rate and energy cost are halved. Combined with 8-bit quantization, feature storage shrinks by 75% without accuracy loss.

For the course's research question, this implies that few-shot gesture learning is feasible on embedded devices: (i) the compact feature space is preserved at low sampling rates, (ii) memory and power constraints are respected, (iii) gesture distinctions (*SHAKE* vs. *CIRCLE* vs. *TILT*) remain clear.

This validates 50 Hz as a robust operating point for embedded gesture recognition, supporting future work on on-device learning with only a handful of examples.

## IV. Issues

The following practical issues were observed during data collection and analysis: (i) transition segments between gestures produced minor misclassifications; (ii) gravity removal via high-pass filtering can distort very low-frequency tilt; (iii) potential aliasing risk without anti-alias filtering prior to decimation; (iv) timestamp jitter from USB streaming introduces small window misalignments; (v) class imbalance across recorded segments affects rule threshold selection.

### A. Readiness for Embedded Demonstration

The current implementation performs feature extraction and rule-based classification offline in Python using logged data. A microcontroller-ready implementation of the same features and thresholds is prepared for the Raspberry Pi Pico, with real-time streaming at 50 Hz. The demonstration is ready on device using fixed thresholds; on-device few-shot adaptation is left as future work.

## V. Code Availability and Build/Run Instructions

### A. Firmware (Raspberry Pi Pico)

The firmware is contained in the `project/` directory, organized into `src/`, `include/`, and `lab_algorithms/` modules. The build system is based on CMake and the Raspberry Pi Pico SDK.

To build the UF2 firmware file:

```
cd project
mkdir build && cd build
cmake .. -DPICO_SDK_PATH=../pico-sdk
make -j
```

The generated UF2 file is located in the `project/build/` folder. It can be flashed to the Raspberry Pi Pico by holding BOOTSEL while connecting the device via USB and copying the UF2 file to the mounted volume.

### B. Logging from the Pico

Once flashed, the Pico streams IMU samples and extracted features over USB CDC. Inside the provided Docker container `MLES_labs`, the serial device can be identified with:

```
ls /dev/cu.usbmodem*
```

A new log file can then be created as follows:

```
mkdir -p logs
cat /dev/cu.usbmodem1301 | tee logs/test_01.csv
```

The output is continuously stored in the `logs/` folder as a CSV file.

### C. Analysis and Reproduction of Figures

Data analysis is performed entirely in the `analysis/` directory. The Jupyter notebook `hw1_analysis.ipynb` parses the CSV logs, computes features, applies the classifier, and generates all figures and tables included in this report.

To run the analysis:

```
cd analysis
jupyter notebook hw1_analysis.ipynb
```

Executing the notebook reproduces the results and regenerates the figures stored in `analysis/figures/`.

## VI. Conclusion

It is demonstrated that IMU-based gesture classification is robust to halving the sampling rate and to feature quantization. Both steps significantly reduce system cost while preserving accuracy. This directly addresses the course theme of enabling low-power few-shot learning on embedded hardware.

## References


[1] "Methods — accelerometer documentation (UK Biobank accelerometer analysis)," online manual, https://biobankaccanalysis.readthedocs.io/en/latest/methods.html.

[2] A. Walmsley *et al.*, "Impact of Reduced Sampling Rate on Accelerometer-based Physical Activity Monitoring and Machine Learning Activity Classification," 2020. Available: https://www.researchgate.net/publication/346430018_Impact_of_Reduced_Sampling_Rate_on_Accelerometer-based_Physical_Activity_Monitoring_and_Machine_Learning_Activity_Classification.

[3] J. Dai *et al.*, "Human Activity Recognition from Accelerometry, Based on a Radius of Curvature Feature," *Mathematics*, vol. 29, no. 5, 2021. https://www.mdpi.com/2297-8747/29/5/80.

[4] "The impact of sampling rate on the classification accuracy using FNSW and FOSW," figure resource