GESTURE DETECTION SYSTEM

SIRIGUDI MIDHUSH(220150024)

1.Data & Machine Learning for Gesture Recognition

Gesture definitions

We collected three simple gestures—"circle," "X," and "Y"—using an ESP32 sensor. We chose these because they're easy to generate consistently (unlike real-machine anomalies) yet still let us explore MicroPython and TinyML on constrained hardware.

Circle: a roughly 800–1,000 ms circular motion.

X and Y: straight strokes along the sensor's X- or Y-axis, lasting about 400–600 ms each.

2. Data collection

To capture raw sensor streams, we ran this MicroPython snippet on the ESP32 at a 10 ms interval (100 Hz)—the fastest stable rate we could achieve:

We captured the terminal output to a text file (using VS Code's Terminal Capture), then converted it to CSV for labeling.

3. Labeling & exploratory analysis

Using Label Studio, we tagged each 1,000 ms window with one gesture. Although "X" and "Y" vary in duration (400–600 ms), we gave all gestures the full 1,000 ms span to simplify windowing.

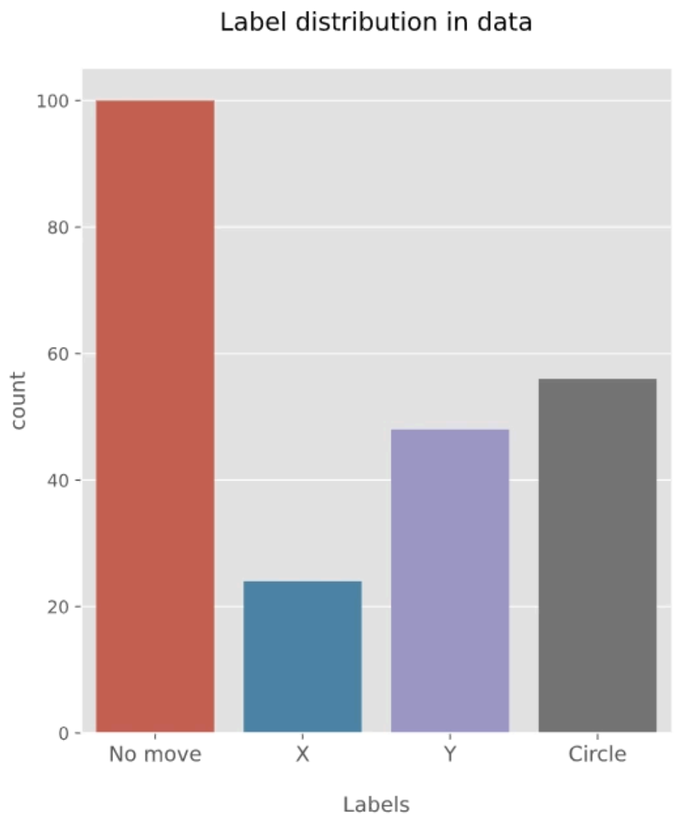
EDA: 3D plots of (X,Y,Z) acceleration and angular velocity reveal distinct clusters for each gesture. Correlation matrices (in our Jupyter notebooks) confirm consistent signal patterns.

4. TinyML requirements

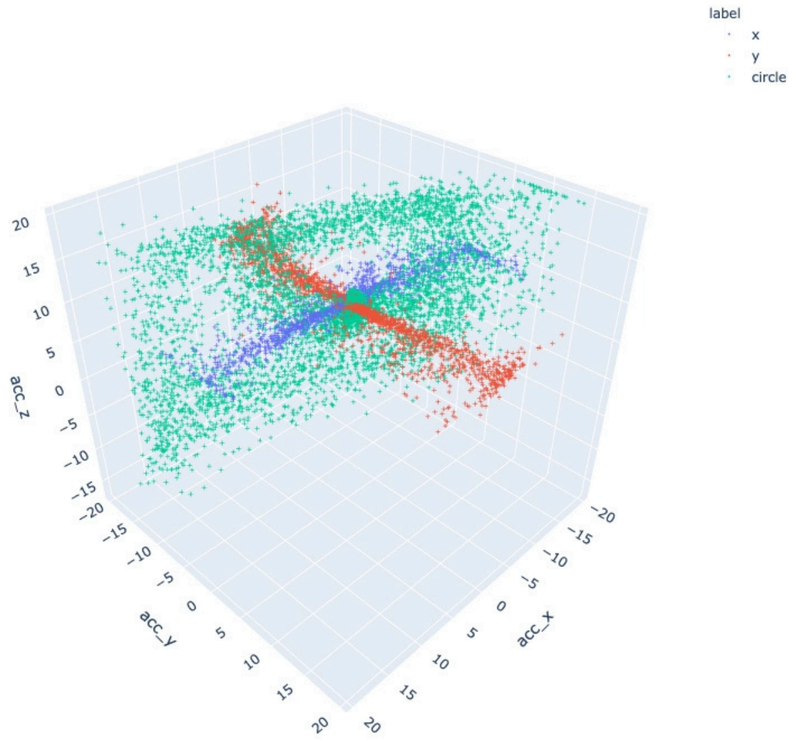
On an ESP32, we must keep:

Inference time $\ll$ 10 ms (our sampling period)

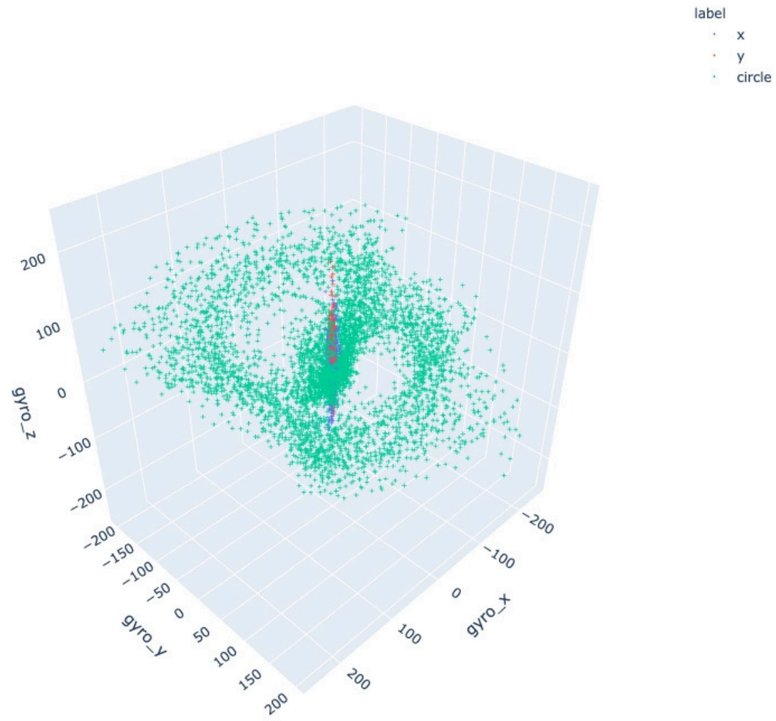
Model size $<$ 20 kB (MicroPython file limits)

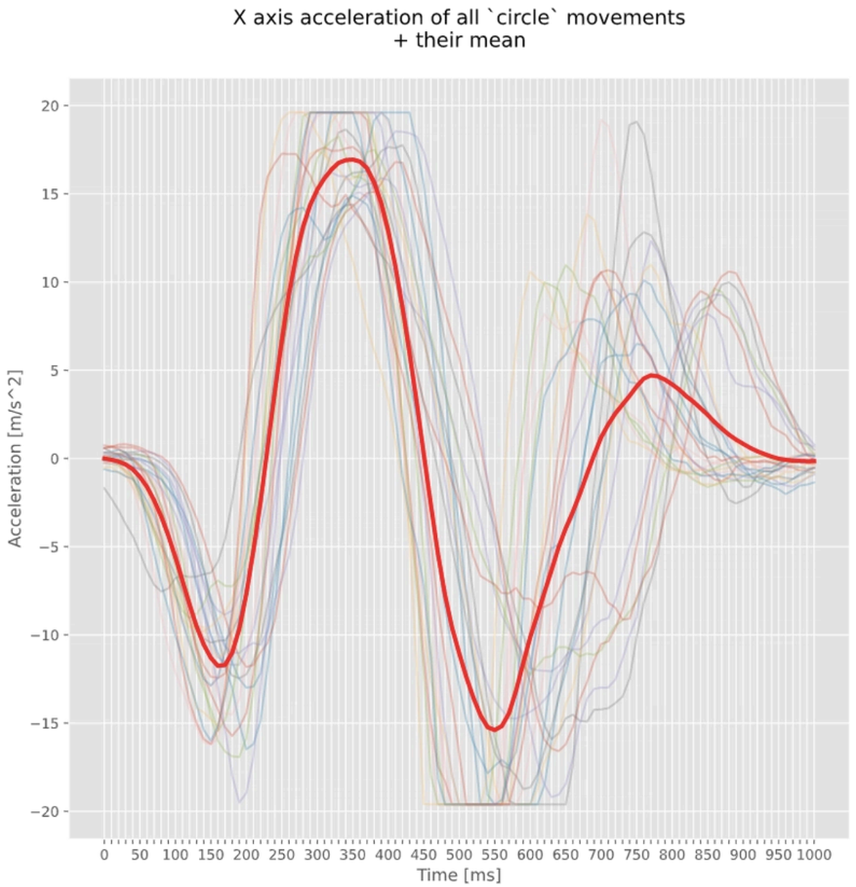


Acceleration 3D plot image



Angular velocity 3D plot image





Note: The cutoff at the top and bottom of the signals is due sensor range which was set to 2G (~19.6 m/s^2).

5. Dataset variants

We created five training datasets to see how well they are working.

Description

1)Baseline: raw labeled windows.

2) Centered X/Y: shift X and Y so the stroke sits in the middle of the 1,000 ms window.

3)Centered + Augmented: Centered gestures with small random shifts added to simulate natural variation and improve model generalization..

4) Centered + SMOTE:Centered gestures with extra fake examples added to balance all gesture types equally..

5) End-aligned X/Y: place X and Y at the end of each 1,000 ms window.This helps us test if the position of the gesture within the window affects how well the model learns."

We also experimented with downsampling to 50 Hz and 20 Hz.

ms	acc_x	acc_y
0	10	10
10	2	2
...	...	...
1000	3	3

acc_x_0	acc_y_0	acc_x_10	acc_y_10	...	...	acc_x_1000	acc_y_1000
10	10	2	2	...	...	3	3

Sampling frequency	Sampling period	Data points (6 signals)
100Hz (as collected)	10 ms	606
50Hz (downsampled)	20 ms	306
25Hz (downsampled)	40 ms	156
20Hz (downsampled)	50 ms	126
10Hz (downsampled)	100 ms	66

6. Baseline model results

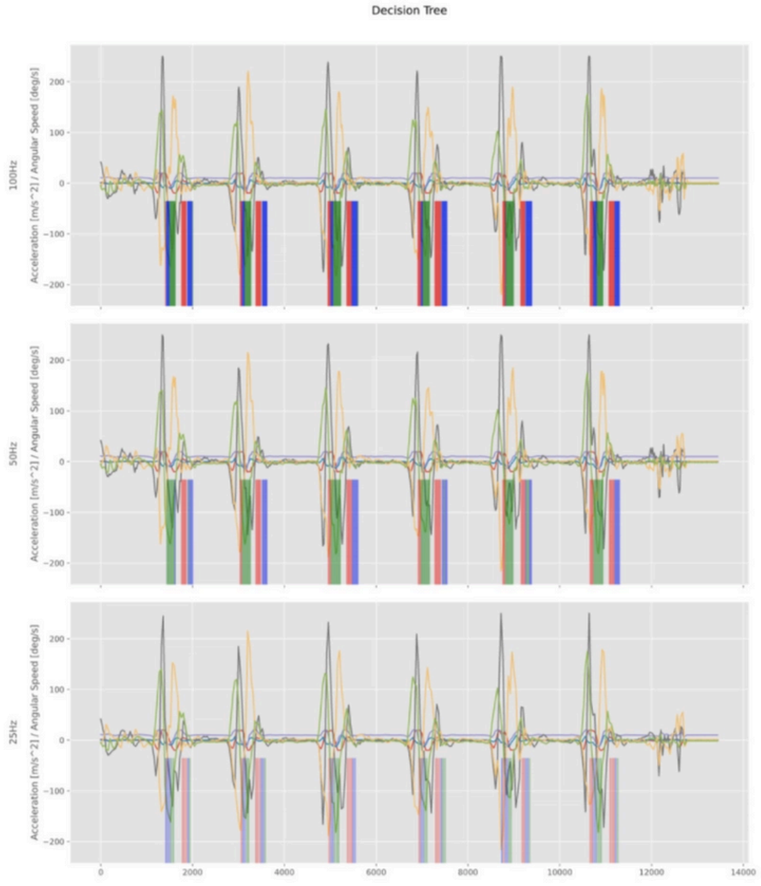
We tried Decision Tree, Random Forest, SVM, Logistic Regression, and Naive Bayes—but settled on Decision Tree and Random Forest because:

Only they converted cleanly to pure-Python (via m2cgen).

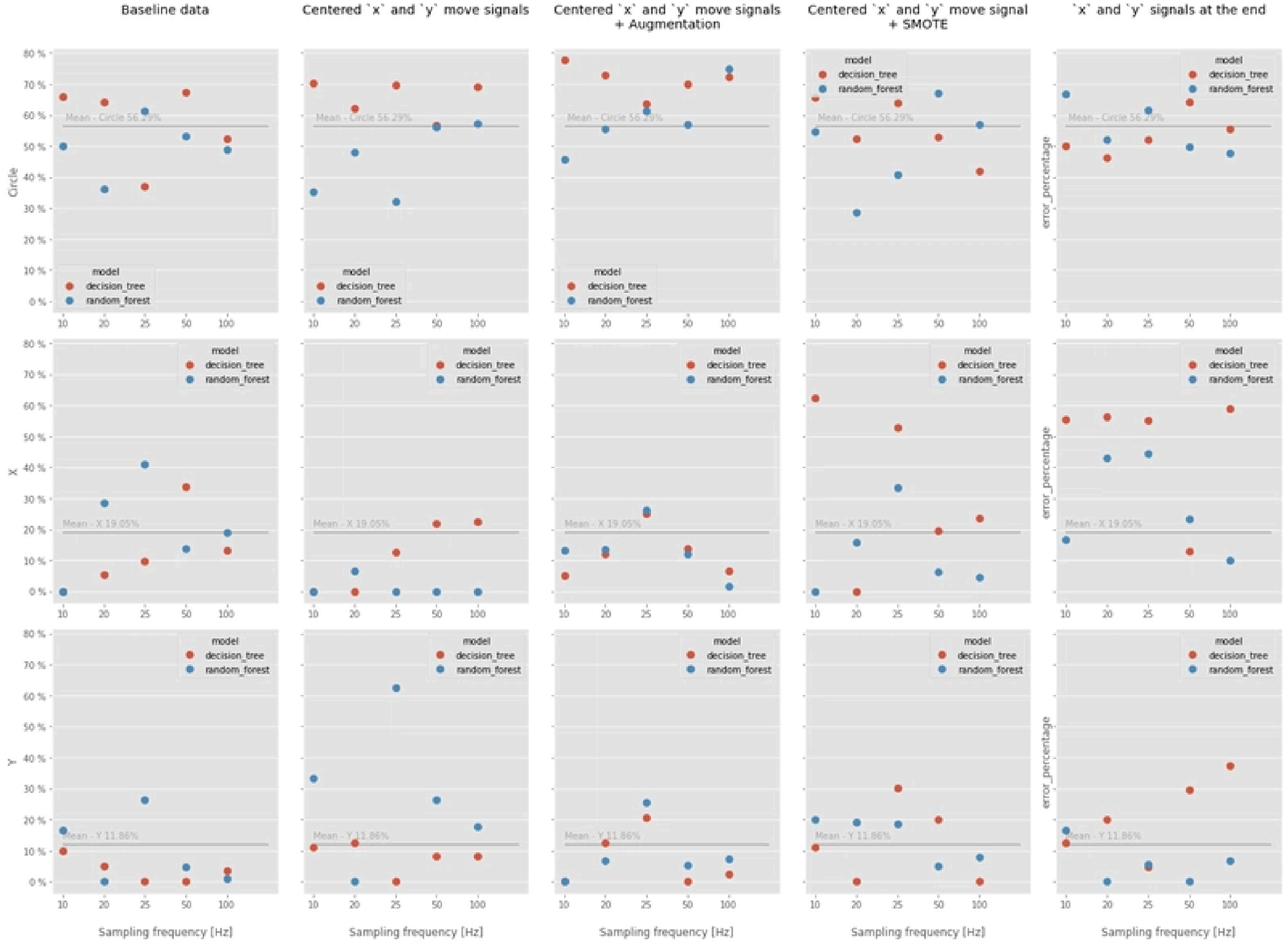
SVM/Logistic Regression were too slow at inference.

Naive Bayes couldn't convert.

All models scored > 95 % accuracy on static windows, but live-data tests (sliding 1,000 ms window, infer every 10 ms) revealed many false or repeated detections.



Label	Equation
Cirle	cirle_error = X+Y / (X+Y+Circle)
X	x_error = Circle+Y / (X+Y+Circle)
Y	y_error = Circle+X / (X+Y+Circle)


Baseline data
Centered `x` and `y` move signals
Centered `x` and `y` move signals + Augmentation
Centered `x` and `y` move signal + SMOTE
`x` and `y` signals at the end
Circle
X
Y
error_percentage
Mean - Circle 56.29%
Mean - X 19.05%
Mean - Y 11.86%
model
decision_tree
random_forest
80 %
70 %
60 %
50 %
40 %
30 %
20 %
10 %
0 %
10
20
25
50
100
Sampling frequency [Hz]

7. Inference-time testing

On an ESP32 at 160 MHz, a Random Forest with 10 estimators runs in ~4 ms—fast enough for 100 Hz sampling. At 240 MHz, it will be even quicker. Since Random Forest = ensemble of trees, any DT that passes will also pass

.

8. Model optimization

We ran a grid search over:

Number of estimators (3–10)

Input channels (e.g. just accelerations vs. accel + gyro)

Sampling rates (20, 25, 50 Hz)

The best optimized model (ID #2) used fewer signals at 50 Hz with 5 estimators. Surprisingly, reducing inputs actually cut false inferences.

9. Debouncing live inference

Even the optimized model still produced multiple votes per gesture. To emit exactly one detection per event, we:

Buffer each inference (timestamp, label).

Wait until $\geqslant$ 9 inferences accumulate (200 ms window at 50 Hz).

Ensure time span $\geqslant$ 450 ms (to filter trailing echoes).

Return the most frequent label in the first 9 votes, then clear the buffer.

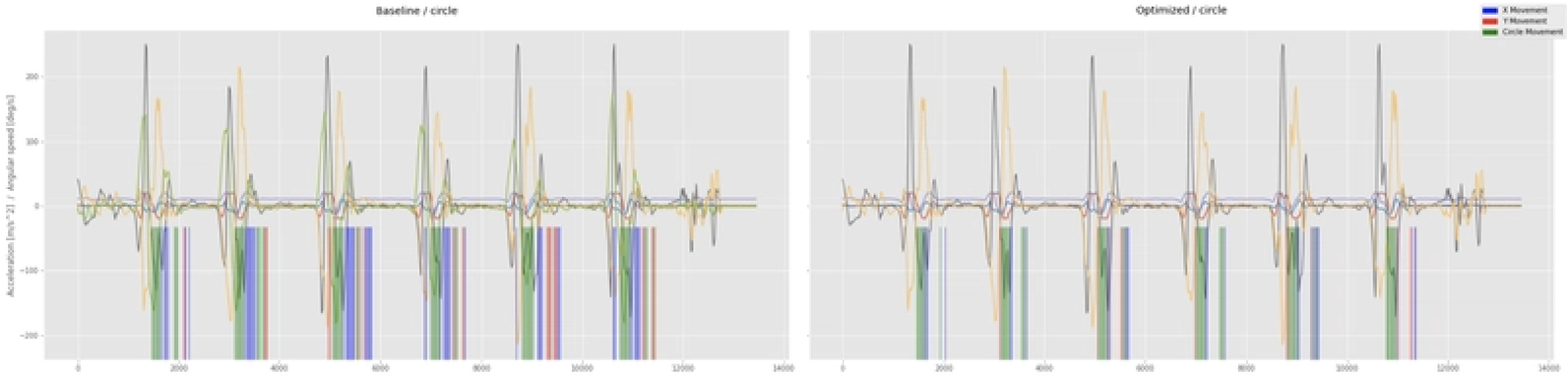
This simple debounce sharply cleans up "circle," "X," and "Y" detections in our live-data validation.

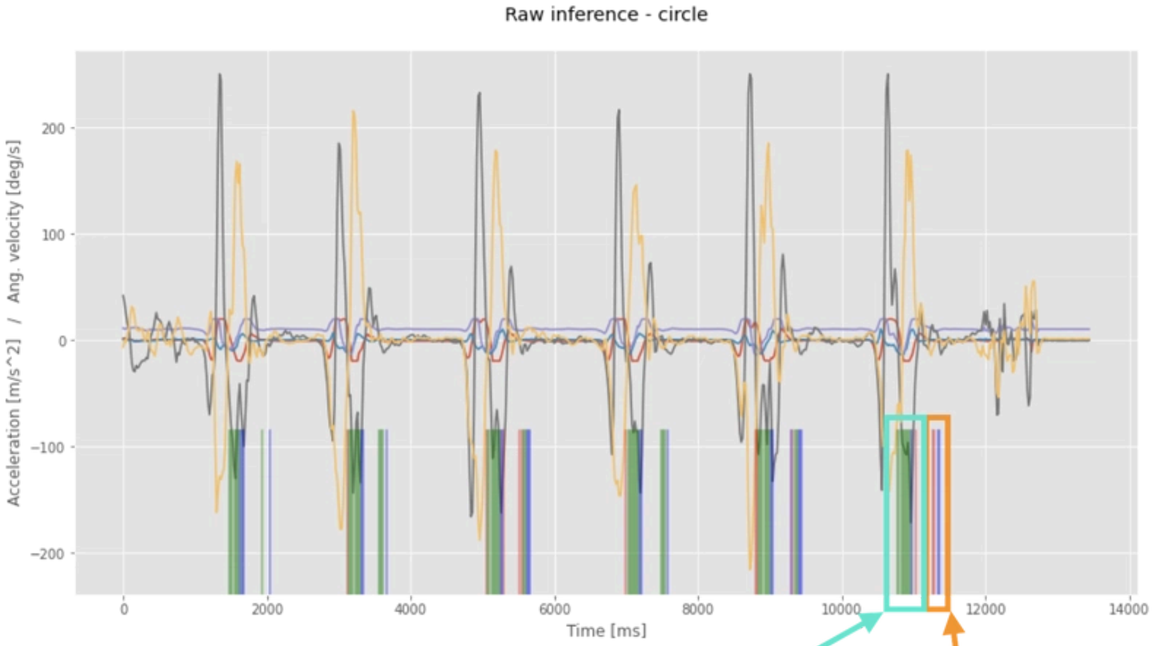
10. Next steps

Experiment with smarter debounce (dynamic windowing).

Try lightweight neural nets (e.g. 1D-CNN) if they fit.

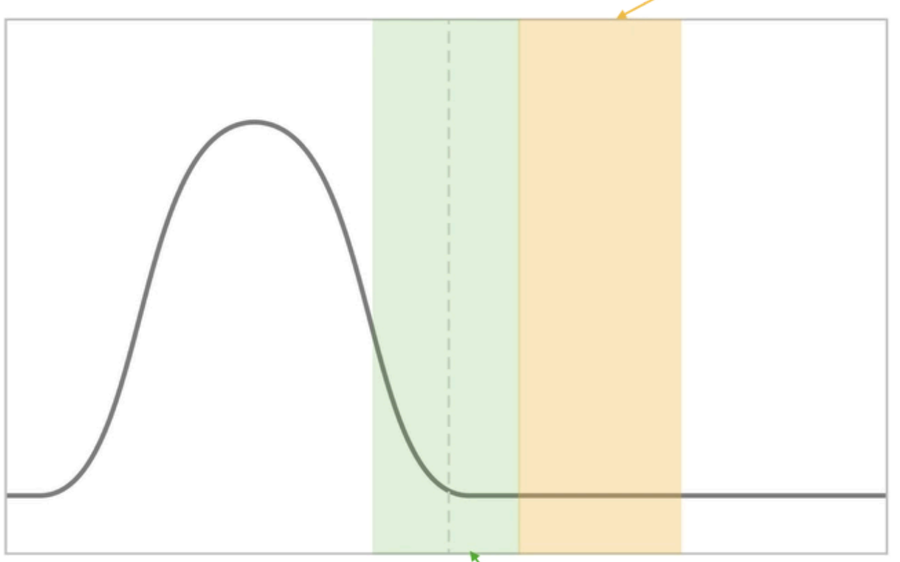
Gather more real-machine anomaly data for transfer learning.





Groups of same (correct) inference results

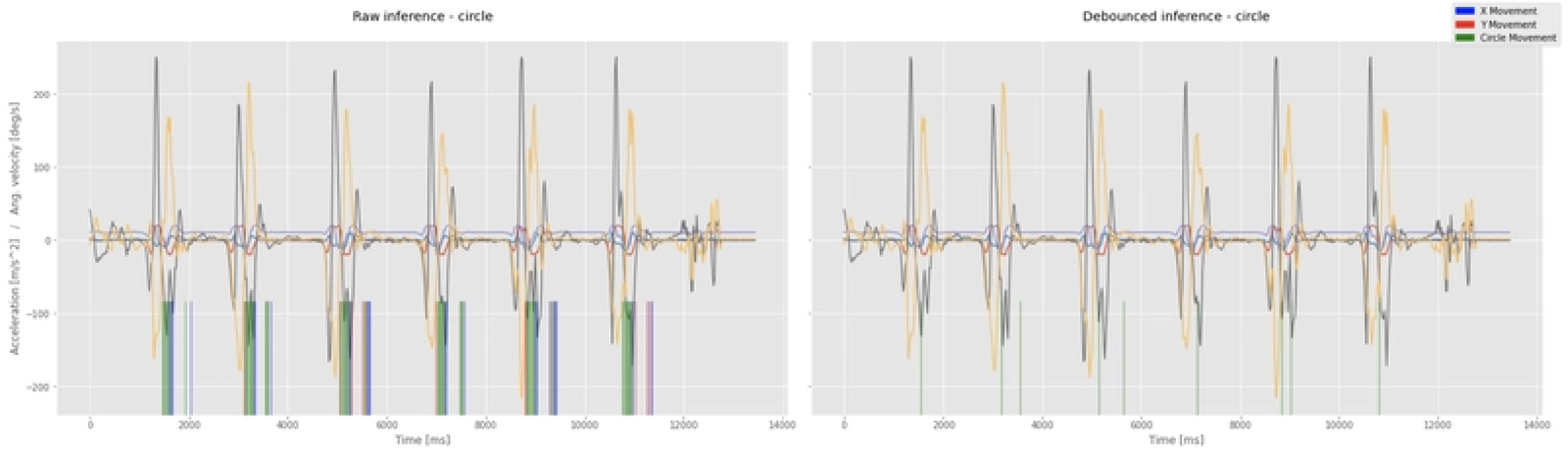
'Trailing inference'



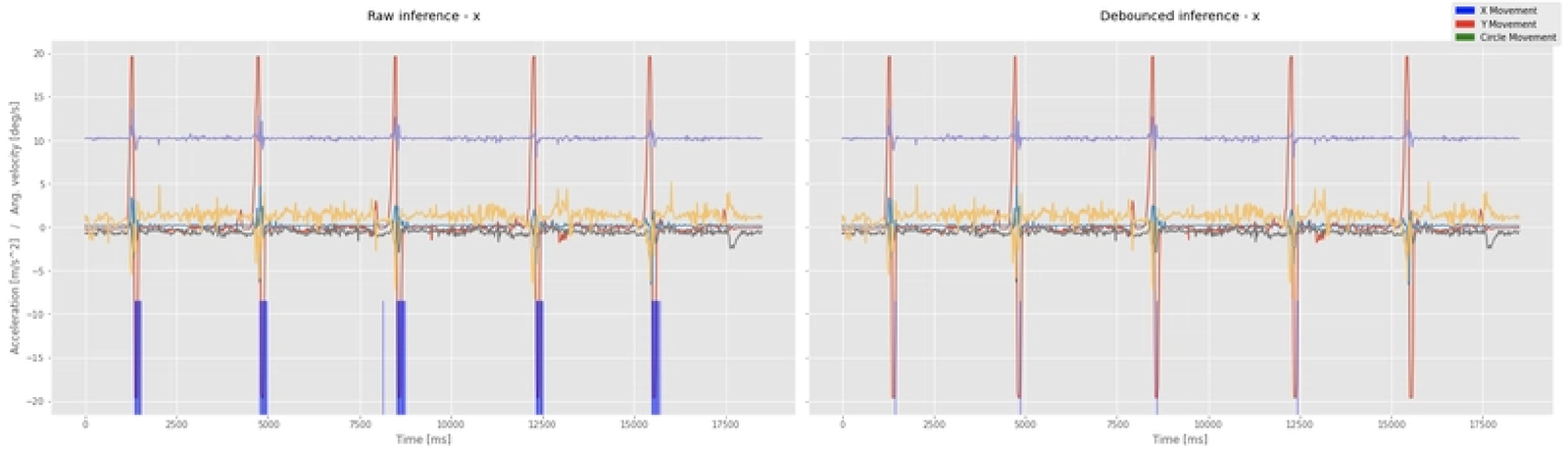
Time diff to filter out 'trailing inferences' (~450ms)

Expected inferences (~ 200ms)

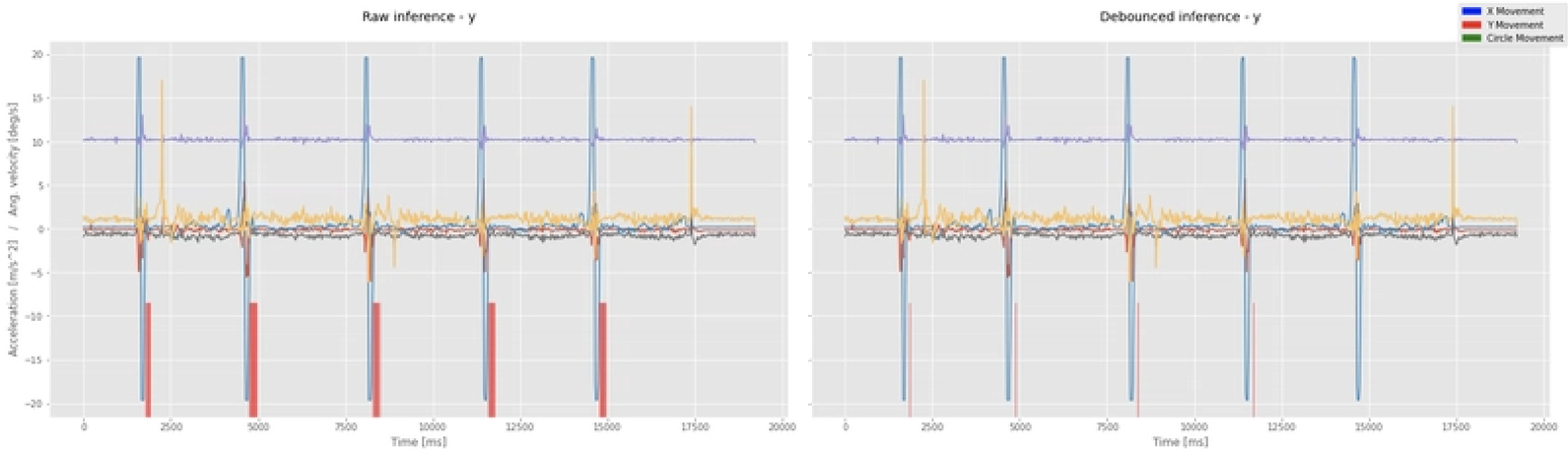
Circle


Raw inference - circle
Debounced inference - circle
X Movement
Y Movement
Circle Movement
Acceleration [m/s^2] / Ang. velocity [deg/s]
Time [ms]

X


Raw inference - x
Debounced inference - x
X Movement
Y Movement
Circle Movement
Acceleration [m/s^2] / Ang. velocity [deg/s]
Time [ms]

Y


Raw inference - y
Debounced inference - y
X Movement
Y Movement
Circle Movement
Acceleration [m/s^2] / Ang. velocity [deg/s]
Time [ms]
Time [ms]

THANK YOU