Title: Real time pre-fire event prediction for integrated mechatronics system using IMU data

Names & student numbers: Stan Coene (r0935348), Karel Van Der Haegen (r0955863

Assigned coach: Dimas

Introduction

Instructions: Briefly recap your motivation, the background of the topic and why it matters. Provide a literature review on the relevant task.

For the past 2 years I (Stan) have been developing an open-source project called “Stinger”. It’s an mechatronics device using advanced technologies in an embedded application. Its ‘use’ is to shoot nerf darts at high speed/firing rates, but to achieve that there is a lot of software / hardware cooperation. We would like to locally run a small model to predict user firing. This will speed up the reaction time of the flywheels by pre-spinning them before the user decides to fire, based on prior movements in gyro and acceleration data. Pre-spinning flywheels reduced fire delay from 150ms to 30ms approximately after a trigger pull.

This predicting of user behavior based on IMU data to reduce latency is not limited to this specific application, so these models can have wider use in handheld mechatronics.

Method

Data

Instruction: Describe the dataset and cover the following elements.

- Dataset: processed_features
- Citation / reference: https://github.com/Project-Stinger/Firmware/tree/MLmodel
- Source / accessibility: public
- Description:
  - Task: classification
  - Number of samples:
  - Number of classes (if classification):
  - Type of features: a typical 1–10-minute recording session generates ~60,000-1.000.000 samples
  - Number of features:
    - Data format: CSV
    - Class balance: Highly imbalanced. The Pre-Fire Event constitutes a very small fraction of the total dataset, as firing is an infrequent action compared to holding, moving, or aiming without firing

Data processing: This is the primary preprocessing step. The raw time-series data is converted into a tabular format suitable for classification models using a sliding-window approach.

Labeling: A Pre-Fire Event label (Class 1) is generated by identifying every instance where the trigger_state column changes from 0 to 1. The 500ms (500 samples at 1000Hz) preceding this trigger pull are then labeled as 1. All other samples are labeled 0.

Feature Engineering: A 100ms (100-sample) sliding window is applied to the raw sensor data to extract 30 features for every sample:

- **Accelerometer** (18 features): To capture linear motion like "snap-aims" and "micro-adjustments," 6 features are extracted from each of the 3 axes: mean, std, min, max, diff_mean (mean of the rate-of-change), and diff_std (standard deviation of the rate-of-change).
- **Gyroscope** (12 features): To capture general rotational movement, 4 features are extracted from each of the 3 axes: mean, std, min, and max.

Finalization: The first 99 rows are dropped (as the 100ms window is not yet full), resulting in a processed_features.csv file. Train/validation/test split: The data is split into training, validation, and test sets (e.g., 70/15/15). Shuffling is set to False to respect the time-series nature of the data, ensuring the model is trained on past events and tested on future, unseen events.

Models

Instruction: Describe the machine learning techniques you have applied in bullet points.
Model 1: logistic regression
Model 2: Random Forest

Evaluation metrics

We focus on the performance for the "Pre-Fire Event" because it represents success. Our top priority is Recall, which measures how many actual shots the model successfully detects early. We also monitor Precision, which measures how often the model's prediction to spin the flywheel is actually correct (to prevent unnecessary spin-ups) . And also Specificity which is the percentage of actual negative events that the model correctly identified.

Preliminary Results

Instruction: Update your obtained results and analysis, at least with model 1. Benchmark with the paper results.

Logistic regression:
After some fine tunning of the code from the lab we got this result (second row of the table), where you can see that our model almost always spun the wheels when necessary, but also did this a lot of times when the user was not about to shoot. I compared this to a paper were they also use logistic regression for detecting if an elderly person had fallen. We can see that there model reaches way better results than ours, but it does miss a couple of positives. The paper did not give a precision, but our model got one of 27%.

Metric	Our Result	Benchmark Result (Fall Detection)

Accuracy	87%	98.17%
Recall	99%	87.97%
Specificity	87%	98.98%

This 27% precision clearly says that it really does not want to miss a single shot so it just guesses that the user is going to shoot a lot and because our data is pretty unbalanced the accuracy is still decent. So it is clear that the logistics regression model is to simple to correctly predict the user to shoot with this large amount of possible movements.

Next steps

For the implementable model we want to use random forest so it can run locally on our MCU (RP2040). We've also trained a NN (simple MLP with 2 hidden layers) with pytorch but are still making changes to it's architecture to see what works best. Same with the random forest model. The behavior heavily depends on the quality of the dataset. Datasets with lots of random information and false negative events where a user uses the gun in standard unpredictable ways. Like usually you hold the gun still and then shoot so then there's not much information for the model to train on. With random forest we've noticed a larger window size from 100ms can also greatly increase accuracy but not in all cases. We have now recorded a larger dataset of semi-consistent shooting, aligned with normal operation but still enough 'predictable' data before a data event. We are aiming to get a recall of above 80% (sometimes it is unrealistic to be able to predict a shot) and keep false positives as low as possible.

Comments to the coach (optional)

Instructions: feel free to note your questions.

Reference

Instruction: Cite your dataset and any reference you referred above.

https://github.com/Project-Stinger/Firmware/tree/MLmodel

Otanasap, N., Bangkomkun, P., & Tanantpapat, T. (2023). Pre-Impact Fall Detection System Using Logistic Regression Model: Pre-Impact Fall Detection System Using Logistic Regression Model. *SAU JOURNAL OF SCIENCE & TECHNOLOGY*, *9*(1), 30–43. retrieved from https://ph01.tci-thaijo.org/index.php/saujournalst/article/view/251805

Reminder

Please save your proposal file in pdf and name as:

Milestone_[coach name]_[C/R]_[surname1]_[surname2].pdf

Where:

- coach name = your assigned coach number

- C/R = task type (C = classification, R = regression)
- surname1_surname2 = the last names of both team members

Examples:

- Coach Diwas, Regression, team Janssen & De Smet → Proposal_Diwas_R_Janssen_De Smet.pdf
- Coach Meixing, Classification, team Li & Garcia → Proposal_Meixing_C_Li_Garcia.pdf

Please delete this section in your submission!