TenniSense - Tennis Racket Sensor - IoT Final Project

Yehonatan Peisakhovsky*
Matan Solomon*
Ofek Glick*

July 24, 2022

Contents

1 Workout Classifier & Step Counter **2**
1.1 Preprocessing of data 2
1.2 Activity Classification 2
1.3 No. of Steps Prediction 2
1.4 App Integration 3

2 Creative project - Tennis Sensor **4**
2.1 Hardware Component 4
2.2 Software Component 4
2.2.1 Frontend - Android App 4
2.2.2 Backhend - Data Analysis 7
2.3 Results 8

1 Workout Classifier & Step Counter

In this section we will elaborate on the first part of the project. This has been integrated into the Android App.

1.1 Preprocessing of data

Before performing any type of learning we begin by preprocessing our data. Given a session with 3 acceleration axes the following actions were taken:

1. Norm Calculation:
   Calculate Norm of 3 axes when Norm was defined to be $\sqrt{X^2 + Y^2 + Z^2}$
2. Anomaly Detection:
   Calculate the $IPR = q_{90^{th}} - q_{10^{th}}$ where q_x represents the x^{th} quantile and remove samples which do not fall into the bound:
   $$[q_{10^{th}} - 1.5 \cdot IPR, q_{90^{th}} + 1.5 \cdot IPR]$$
3. Time Normalization:
   Some sessions do not start at time zero, so the sessions were normalized to always start at time zero.

1.2 Activity Classification

The approach taken to the classification task was to try and use statistical data from a given session and try to extrapolate from that the type of activity.
The following metrics were calculated:

1. Mean value of the X axis.
2. Mean value of the Y axis.
3. Mean value of the Z axis.
4. Mean value of the Norm.
5. Max & Min values of the Norm.

All those features were used to train a Soft SVM model with a regularization value of $C = \frac{1}{100}$. Using this method, we were able to achieve an accuracy of 96%.

1.3 No. of Steps Prediction

Predicting the No. of steps in a session we performed additional data processing. We normalized the Norm by subtracting the mean of the norm, smoothed the data by using a rolling average with a window of 2 (When working on running sessions).

Having done that, we import SciPy's "Peak Detection" function in order to find all local maxima of the recorded session.
When using SciPy's "Peak Detection" function we had to adjust two hyper parameters.

- Cutoff - A certain value which determines the minimum value a point can have to be considered a peak. This is in order to not detect small local maxima during "quiet" periods when there are not steps being taken. In our algorithm this value was set to be $\max(Norm.Std, 1)$ for walking and $\max(Norm.Std/2, 1)$ for running. The reason the cutoff for running and walking is different is because running is a faster activity,
  which results in a more frequent change of acceleration which increases the standard deviation.

- Distance - An integer value determining the minimum no. of samples between two detected peaks. This value was set to 4 for walking and 2 for running. The values were chosen under the assumptions that steps took longer when walking as opposed to running and therefor spread wider apart.

1.4 App Integration

The functionality described above has been added into our app. From the home-screen of the app, one needs to choose the "Part 1 - Analyzing Data" option.
From there, specify the file to analyze
and receive a graph of the acceleration norm over time, estimated steps count and predicted activity.

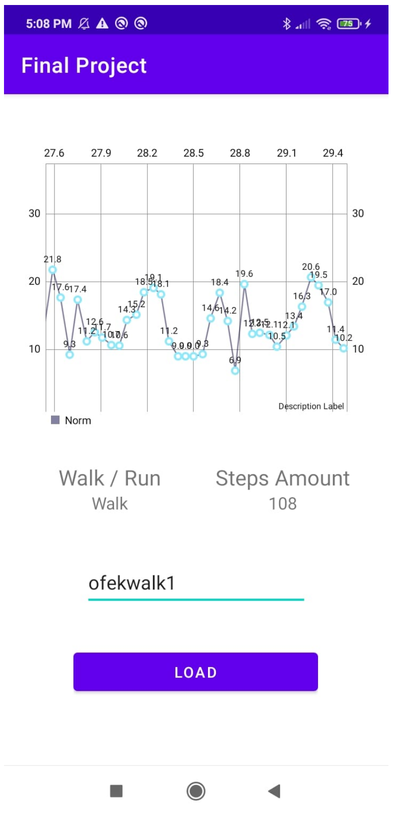

Figure 1: Screenshot of the app

2 Creative project - Tennis Sensor

The product in development was a sensor for tennis rackets consisting of an IMU attached to the throat of the racket and an accompanying app to deliver statistics to the player regarding their athletic abilities.

The intended use of the product was for the player to record a "session" of himself/herself playing tennis and the tennis player would interact with the app to receive statistics regarding the recorded session.

2.1 Hardware Component

The hardware of the project consists of the following:

1. Tennis Racket
2. LSM303DLHC Accelerator & Magnetometer
3. L3GD20 Gyroscope
4. 9V Battery
5. ESP32 Arduino

Components 2-5 were attached via zip ties to component no. 1. See figure 2.

2.2 Software Component

The software component consists of a frontend Android App and a backend python script. The Android App serves as an interface with the sensor and allows the user to utilize the backend python scripts which performs an analysis of a tennis session and outputs detailed statistics regarding the tennis session.

2.2.1 Frontend - Android App

To use the interface with the sensor, one must choose the option "Part 2 - Our System". From there the user is transferred into the main screen of the app. As seen in figure 3, the user is then given an interface to record his/hers tennis sessions.
Before the session could begin, the user first needs to connect to the IMU sensor via Bluetooth. The app will not allow a recording to begin without a Bluetooth device chosen. At the bottom right of the screen there is a button with the Bluetooth symbol on it, to indicate to the user that it is where the device would be selected.
After selecting a device a recording could begin. By pressing "Record" the app will start saving the data transmitted from the IMU sensor. The next action to take after "Record" is "Stop", this action will terminate the recording and will give the recording "Last_Recording". After recording a session, the user has multiple options to choose from for his/hers next action, by pressing "Save" the user could save the session under a custom name, by pressing "Reset" the user could discard the data recorded in the session and by choosing "Report" the user could be transferred to a detailed report generated by the app regarding the last recorded session.

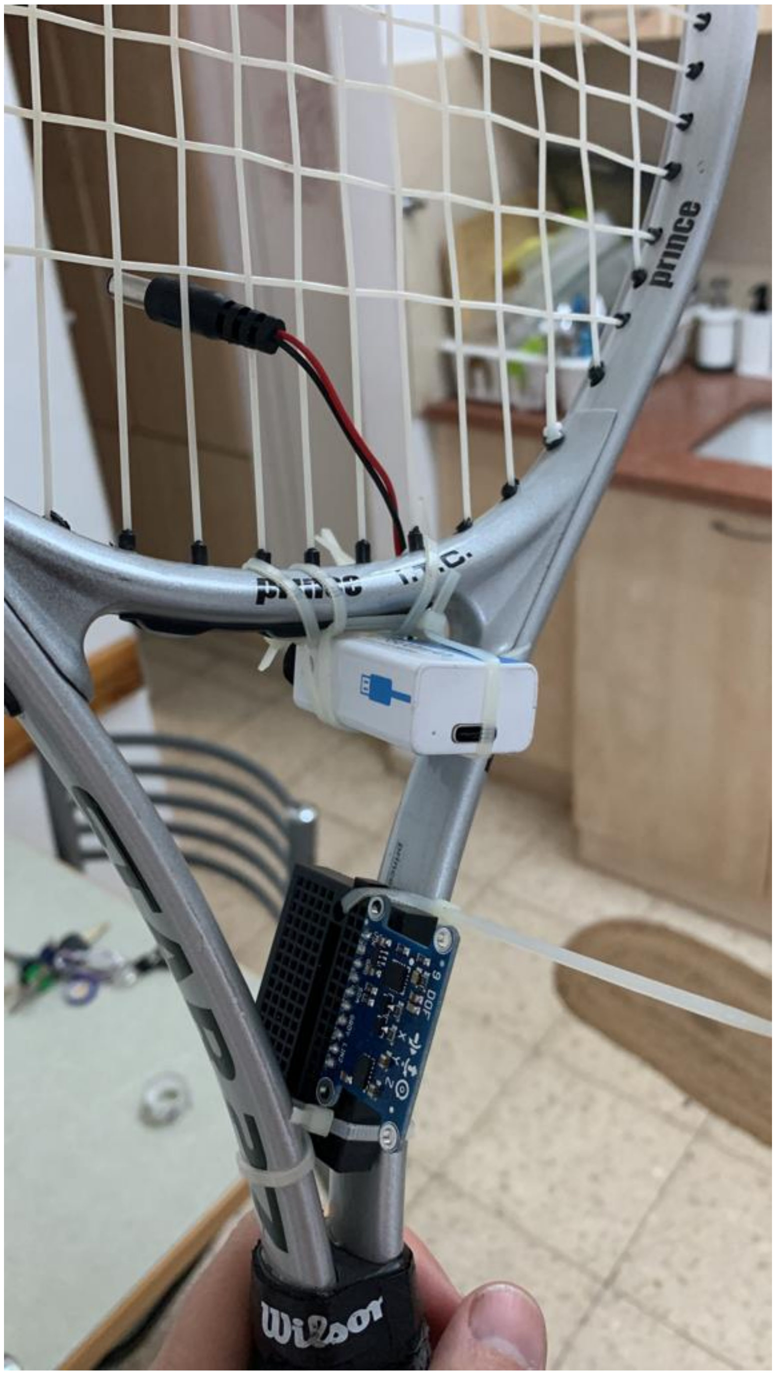

Figure 2: The IMU attached to the racket

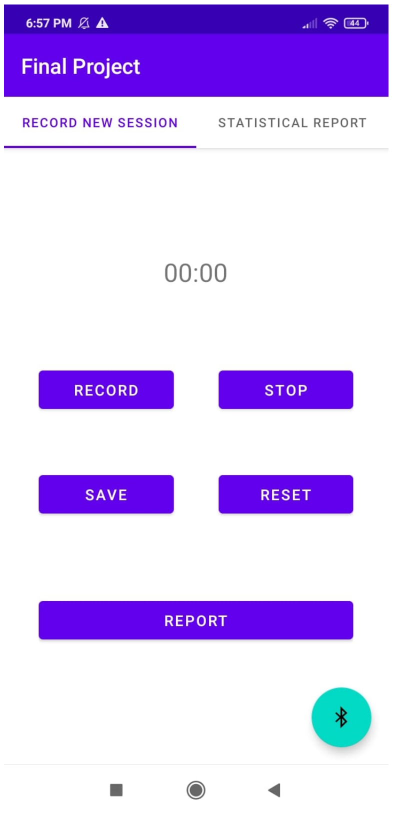

Figure 3: Screenshot of the recording screen

Once the user decides to view the report (either by clicking report or swiping left to the report screen) the screen that will appear is as seen in figure 4
In the recording screen, one could load a previous session by tapping the "settings" icon at the bottom right and specifying the name of the session. In addition, the user could update the app with the users weight to get a personalized calorie measurement (default weight is set to 80 kg). Lastly, the user could email the report by tapping the "send to mail" Button.
The stats shown in the report are:

1. Pie chart specifying the ratio of shot types in the session: Backhands to Forehands.
2. Length of session in minutes
3. No. of shots in the session
4. Max acceleration & speed of the racket
5. Calories burned in the session.

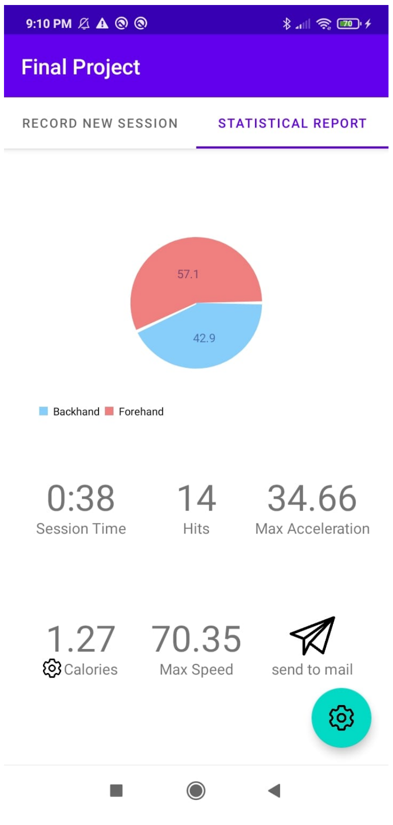

Figure 4: Screenshot of the report screen

2.2.2 Backhend - Data Analysis

We will now present the process done to calculate each of the stats.

1. Calories Burned: We calculate the calories burned at the session using the following formula $Cal = \frac{3.5 \cdot MET \cdot T \cdot Weight}{200}$ where MET (Metabolic Equivalent of Task) is a measurement how strenuous an activity is, Tennis is MET is 5. T is the session time in seconds and $Weight$ is the persons weight in kg.

2. Max acceleration: The maximum value of the norm of the acceleration on all three axes.

3. Session Time: The length of the recording.

4. Max Speed: We calculated the integral of the acceleration on all 3 axes using the "Trapezoidal rule". By assuming the initial speed to be 0 we received the velocity in each axis, and the velocity presented is the norm of the vector of all 3 axes. We attempted to remove the affects of gravity on the sensor using the methods described in this thesis but received poor results and decided to abandon it.

5. No. of Shots: The no. of shots was calculated using the peak detection method described in section 1.3. The Cutoff hyper-parameter was again set to be $\max(Norm.Std, 1)$ and the distance hyper-parameter was set to 25. We tried to implement various filters such as rolling window with various window sizes which did not show any benefit to the results. In addition, we tried using the Butterworth filter using different polynomial orders and different cutoff frequencies but again showed no real benefit to using it.

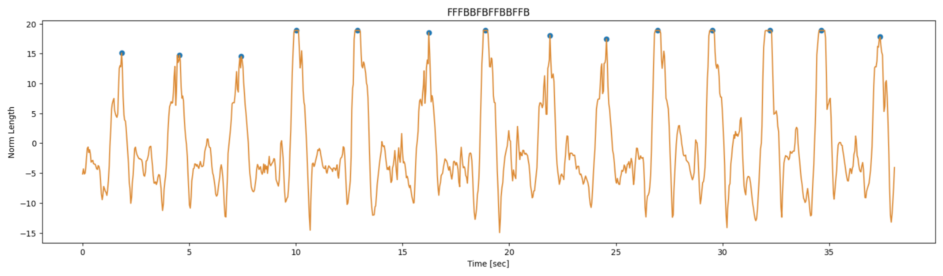

Figure 5: Peak detection at work, finding local maxima with good accuracy

6. Shot classification: The ratio of shots (Forehand vs Backhand). The classifier chose was a Logistic Regression model with L1 regularization to promote sparsity. The features used in the classification were chosen using a package called “tsfresh” which tries to find features who are statistically significant in differentiating between classes. The features that stood out the most were features involving the Y axis of the Gyroscope.

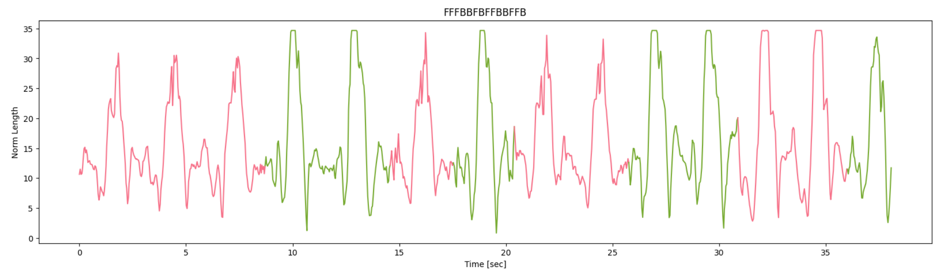

Figure 6: Shot Classification of a session

2.3 Results

After model selection and hyperparameter tuning, we managed to predict with very high accuracy both the no. of shots and the type of shot. We found that assuming that the setting the distance between two peaks to 25 following an assumption that tennis shots usually have some time between them worked really well and the “tsfresh” package managed to find features who managed to classify our shots with very high accuracy.
From multiple tests we performed, the trained models were right on point almost every time.