Project Aim2 Task14

Group5

Institut für Allgemeine Mechanik
RWTH AACHEN UNIVERSITY

Team Introduction

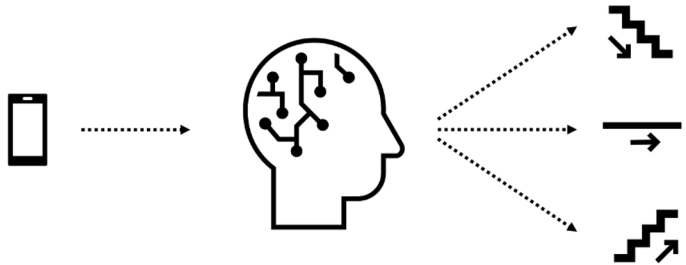
Responsibilities

- Alejandro Delgadillo, Data preprocessing
- Jinyang Yu, Data preprocessing
- Nayoung Ahn, Data preprocessing
- Shuteng Wang, Data preprocessing
- Xiaoyang Liu, Neural network model

Motivation and Aim

Project aim 2

Classification of walking even path, going upstairs or downstairs



Inputs

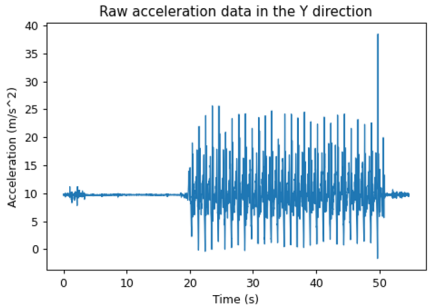
- Data source:
  - Smartphone 3
    Sensor signal: acc and gyr
  - weight and height

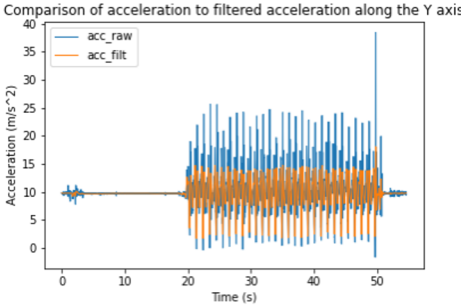
Data Preprocessing - Summary

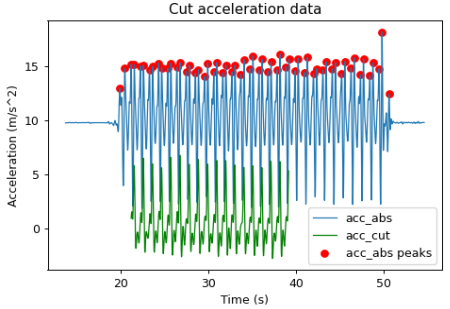
Read csv → Get frequency → Filter noise → Extract motion sequence

anthro_2021.csv

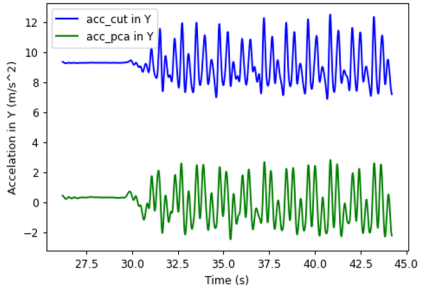
Accelerometer.csv
Gyroscope.csv

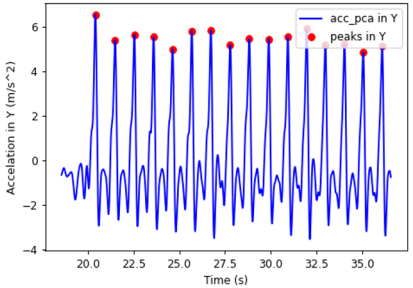


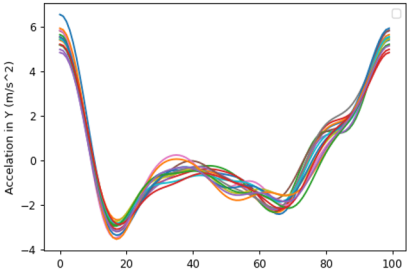


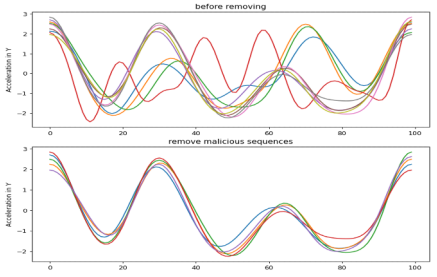


Rotate data → Segment trials into samples → Resample to same frame # → Remove bad sequences





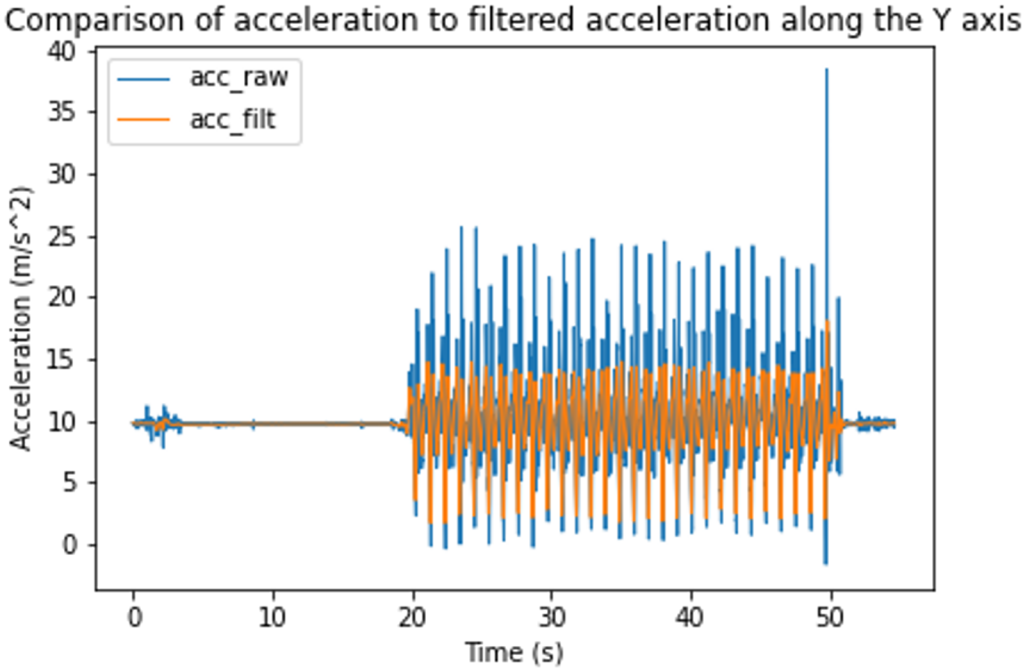




Data filtering

```python
def filter_data(X, Y, Z, fs, fc):
    w = fc / (fs / 2)
    b, a = signal.butter(fc, w, 'low', output='ba')

    X_filt = signal.filtfilt(b,a, X)
    Y_filt = signal.filtfilt(b,a, Y)
    Z_filt = signal.filtfilt(b,a, Z)

    return X_filt, Y_filt, Z_filt
```

- Applied Butterworth low pass filter
- Cut off frequency of 4Hz
  - Avg. walking freq. of a person
  - Fast Fourier Transform



Extract Motion Sequence

Precut data → Get bounds to find peaks → Cut data

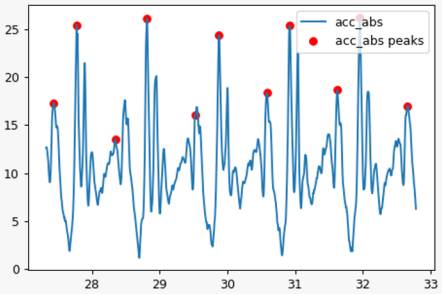
- Precut signals with a precut factor e.g. 0.25
- Precut factor was empirically determined by observing all data

- Get upper and lower bounds to find peaks of the signal

- Starting point: maximum difference between two neighboring peaks
- Duration of e.g. 18s was set manually from the starting pt.

Pre-calculation of parameters

```
def preCalc(X, Y, Z, freq, time):
        start = np.round((0.55 * len(time))).astype(int)
        end = np.round((0.65 * len(time))).astype(int)
        X0 = X[start:end]
        Y0 = Y[start:end]
        Z0 = Z[start:end]
        time0 = time[start:end]

        xyzset = np.column_stack([X0, Y0, Z0])
        xyz_abs = np.linalg.norm(xyzset, axis=1)
        mean = np.mean(xyz_abs)

        peaks, _ = signal.find_peaks(xyz_abs, height=(mean, mean + 50), distance=freq/3)
        heightUpp = np.max(xyz_abs[peaks])
        heightLow = np.median(xyz_abs)

        return heightUpp, heightLow
```



- Get upper and lower bounds of the height signal to find peaks
  - to avoid loss of characteristics of different data
  - used median and max. frequency of the initial data (55% - 65%) to determine the peaks

Extract Motion Sequence

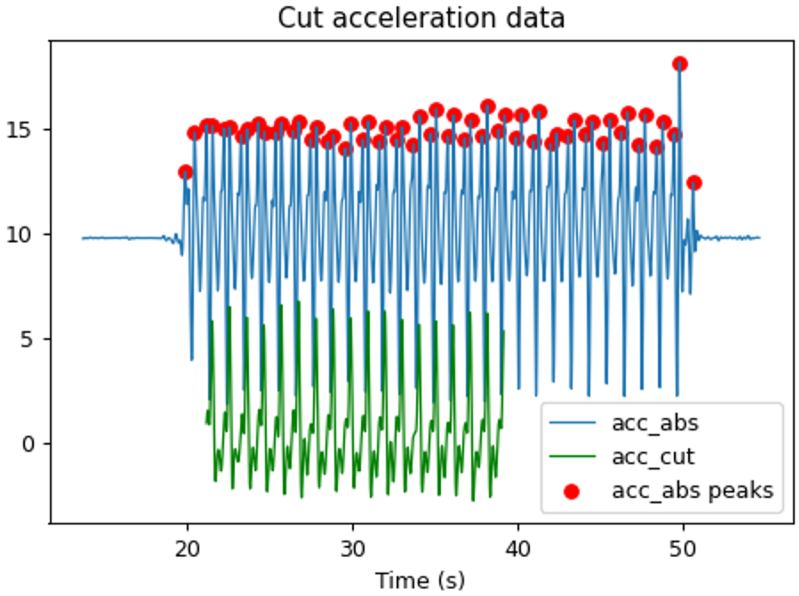
Precut data → Get bounds to find peaks → Cut data

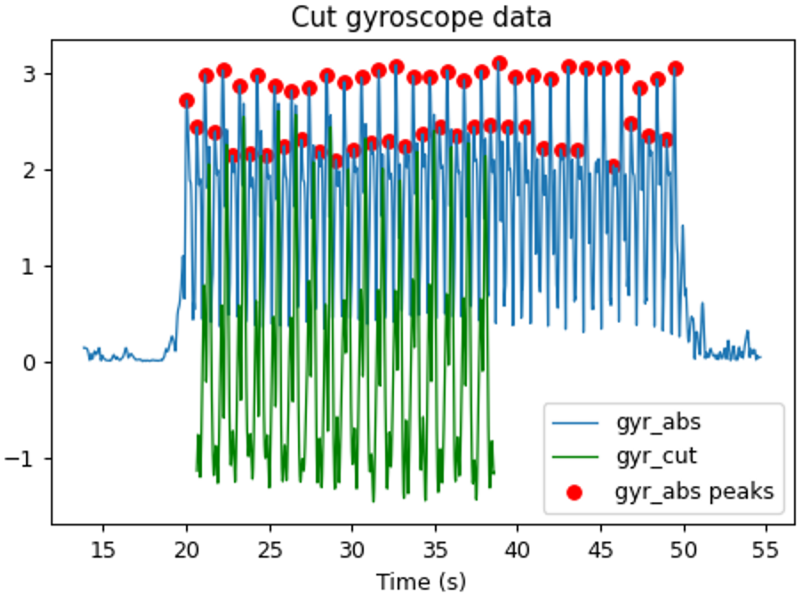
- Precut signals with a precut factor e.g. 0.25
- Precut factor was empirically determined by observing all data

- Get upper and lower bounds to find peaks of the signal

- Starting point: maximum difference between two neighboring peaks
- Duration of e.g. 18s was set manually from the starting pt.

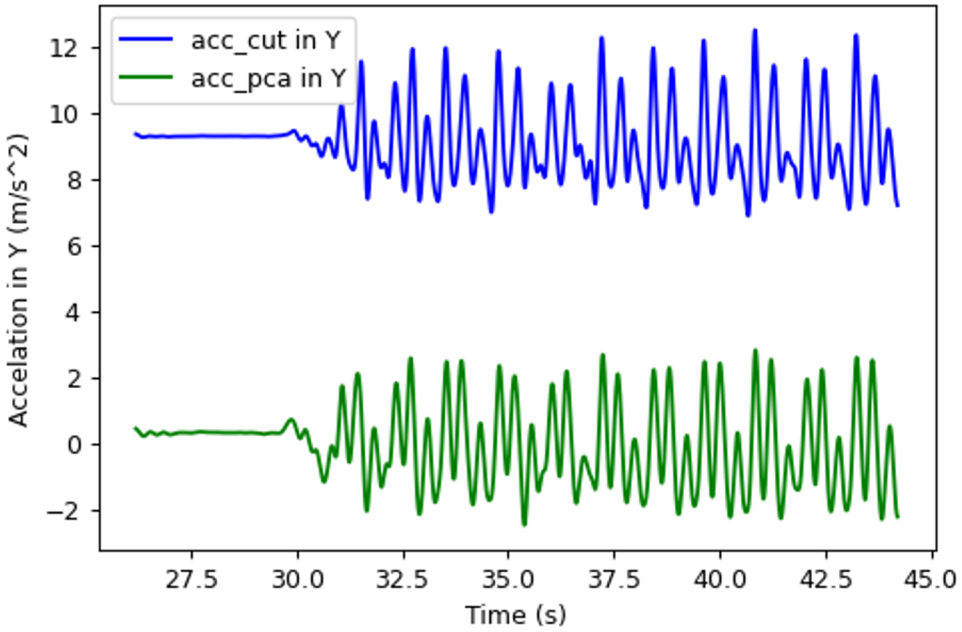
Cutting of accelerometer and gyroscope data





Rotate data using PCA

```
def pca_data(data_cut):
    pca = PCA(n_components=3)
    data_pca = pca.fit_transform(data_cut)
    x_pca = data_pca[:, 0]
    y_pca = data_pca[:, 1]
    z_pca = data_pca[:, 2]

    return x_pca, y_pca, z_pca
```

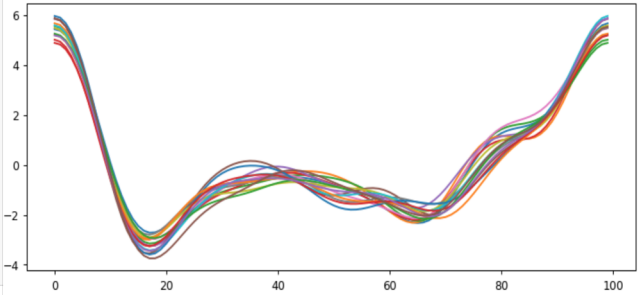


Resample to same frame

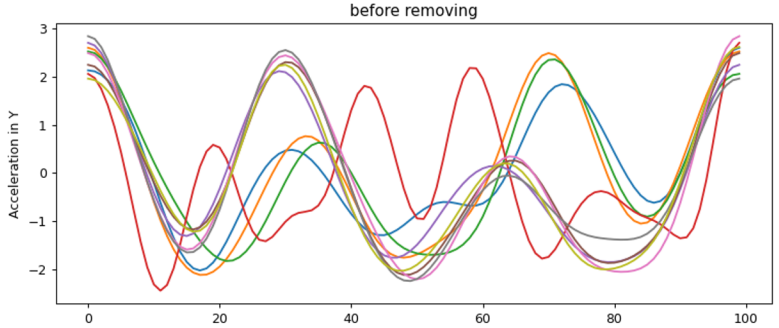
```python
def resample_dataA(a_pca0, time_cutA0, sampling_frequency1, heightUppA, heightLowA):
    time_cutA, indx = np.unique(time_cutA0, return_index=True)
    a_pca = a_pca0[indx]
    peaks_a, _ = signal.find_peaks(a_pca, prominence=0.2 *  (heightUppA - heightLowA), \
                                   height=(heightLowA, heightUppA), distance=sampling_frequency1)
    num = 100
    sampleA = np.zeros((len(peaks_a) - 1, num))
    timelineA = np.zeros_like(sampleA)
    for i in range(len(peaks_a) - 1):
        start = peaks_a[i]
        end = peaks_a[i + 1]
        f = interpolate.interp1d(time_cutA[start:(end+1)], a_pca[start:(end+1)], kind='cubic', axis=- 1)
        timelineA[i] = np.linspace(time_cutA[start], time_cutA[end], num)
        sampleA[i] = f(timelineA[i])

    return sampleA, timelineA
```

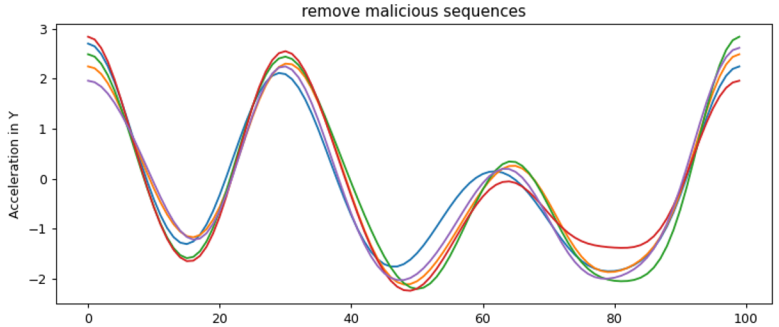
```python
def resample_dataG(g_pca0, time_cutG0, sampling_frequency2, heightUppG, heightLowG):
    time_cutG, indx = np.unique(time_cutG0, return_index=True)
    g_pca = g_pca0[indx]
    peaks_g, _ = signal.find_peaks(g_pca, prominence=0.2 * (heightUppG - heightLowG), \
                                   height=(heightLowG, heightUppG), distance=sampling_frequency2)
    num = 100
    sampleG = np.zeros((len(peaks_g) - 1, num))
    timelineG = np.zeros_like(sampleG)
    for i in range(len(peaks_g) - 1):
        start = peaks_g[i]
        end = peaks_g[i + 1]
        f = interpolate.interp1d(time_cutG[start:(end+5)], g_pca[start:(end+5)], kind='cubic', axis=- 1)
        timelineG[i] = np.linspace(time_cutG[start], time_cutG[end], num)
        sampleG[i] = f(timelineG[i])

    return sampleG, timelineG
```

- Find all peaks after PCA and resample between neighboring peaks
- Each walking cycle in the sequence was resampled to 100 frames per cycle
- The figure shows acceleration in the Y direction



Remove malicious sequences

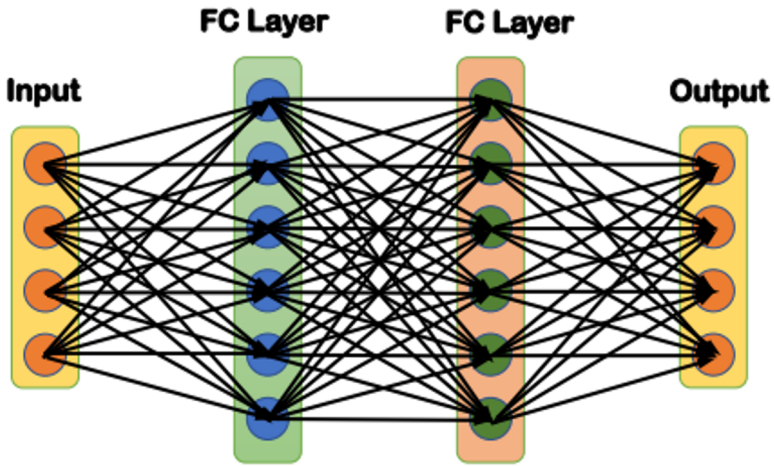




- Using the mean of absolute values of the sequences and the standard deviation of the population.

$$mean_{pop} - (2 \cdot \sigma) < mean_{sample} < mean_{pop} + (2 \cdot \sigma)$$

Neural Network Architecture



Our initial neural network architecture

Input	Input Shape = (8,100)
FC layer	**64 neurons, activation =‘elu’**
FC layer	**32 neurons, activation =‘elu’**
Output	**3 outputs, activation =‘softmax’**

5-fold cross-validation and tuning process

Tab1. Performance with deeper network

2 FC layers	3 FC layers
88.76%	90.12%

Tab2. The effect of input normalization (3FC layers)

Without normalization	With normalization
90.12%	89.24%

Tab3. The effect of weight/height information

Without Height/Weight	With Height/Weight
94.86%	90.12%

Validation process

- Choose 10% of the dataset as test set
- Cut the rest 90% of the dataset into 5 folds
- Use 5-fold cross validation to experiment, compute avg. accuracy and compare

Some results

- 4 + layers did not show improvement.
- Input normalization has minimal affect
- Batch normalization even reduces the accuracy.
- The information of height and weight seems to be redundant.

Final Test

Our final neural network architecture

Input	Input shape = (6,100)
FC layer	64 neurons,activation='elu'
FC layer	32 neurons,activation='elu'
FC layer	16 neurons,activation='elu'
Output	**3 outputs, activation = 'softmax'**

- The 5-fold validation accuracy for our final neural network is 94.86%.
- The accuracy on the test set is 84.15%.
- The accuracy with the information of height and weight on the test set is 79.24%.

Conclusion

- We could classify and predict the walking patterns with a good degree of accuracy from just a well processed smartphone data on a simple neural network architecture.
- Anthropometric data had minimal contribution to the accuracy of the prediction.
- Hence we deduced that acceleration and gyroscope data is most important when determining gait

Discussion

Preprocessing:

- Despite several instances of inconsistent or incomplete raw data we were able to isolate most of those cases and produce a clean dataset to train the neural network.
  - Accurately extracting the motion sequence from raw data was the most challenging preprocessing step.
- When removing the malicious sequences, x, y and z axes were examined independently. This means that for one subject, the samples may include parts from different time frames/ walking cycles. We did this to keep the maximum possible number of samples.

Neural Network:

- Do we really need a neural network to classify the trajectory from accelerometer and gyroscope data?
  - In what ways is a neural network more effective than conventional estimation method?
- In the future, could also try:
  - Recurrent neural network like LSTM
  - Convolutional neural network using rectangular convolution kernel

Thank you for your attention!

Institut für Allgemeine Mechanik
RWTH AACHEN UNIVERSITY