Emotiv-Kit application for 'mind'-controlling mouse pointer

Semester Project for SENSIG course 2022/23

Katherine Sirois, Lorette Dousy, Peter Tibenský

June, 2023

Prof. Dr.-Ing. Andreas König

Overview

1. Introduction
   - *Motivation*
2. Project Steps
   - *Data Acquisition*
   - *Feature Selection*
   - *Spike Detection*
   - *Feature Extraction*
   - *Classification*
   - *Results*
3. Conclusion

Motivation

In future of immersive tech, brain waves scanning technology (EEG) can be used to process simple tasks, freeing up the use of hands for more complex procedures.

Our application showcases the potential of EEG in developing brain-controlled interface for basic movement.

Project Steps

- Raw data acquisition
- Data decryption
- Data manipulation
- Feature extraction

Training (offline)

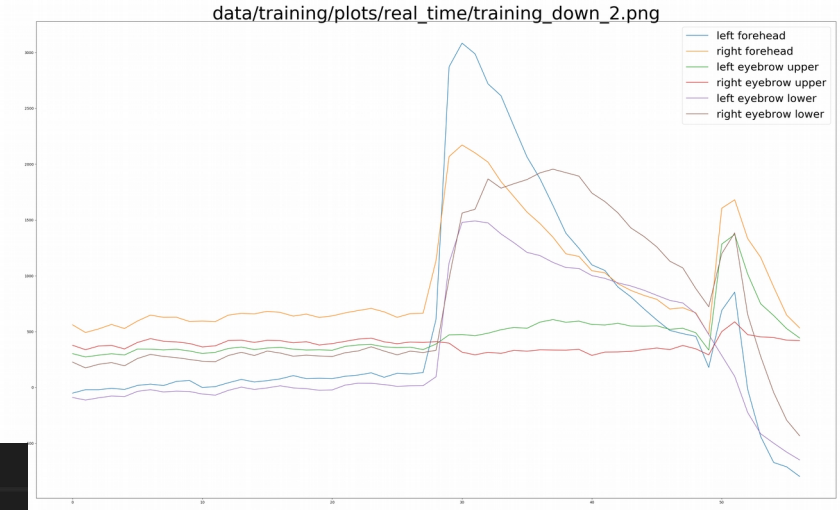
- Plotting
- Averaging

Testing (online)

- Feature matching
- Classification

Data Acquisition

- Initial challenges
  - Paid access
  - Deprecated library
  - Competing resource
  - Non-functional headset and sensor



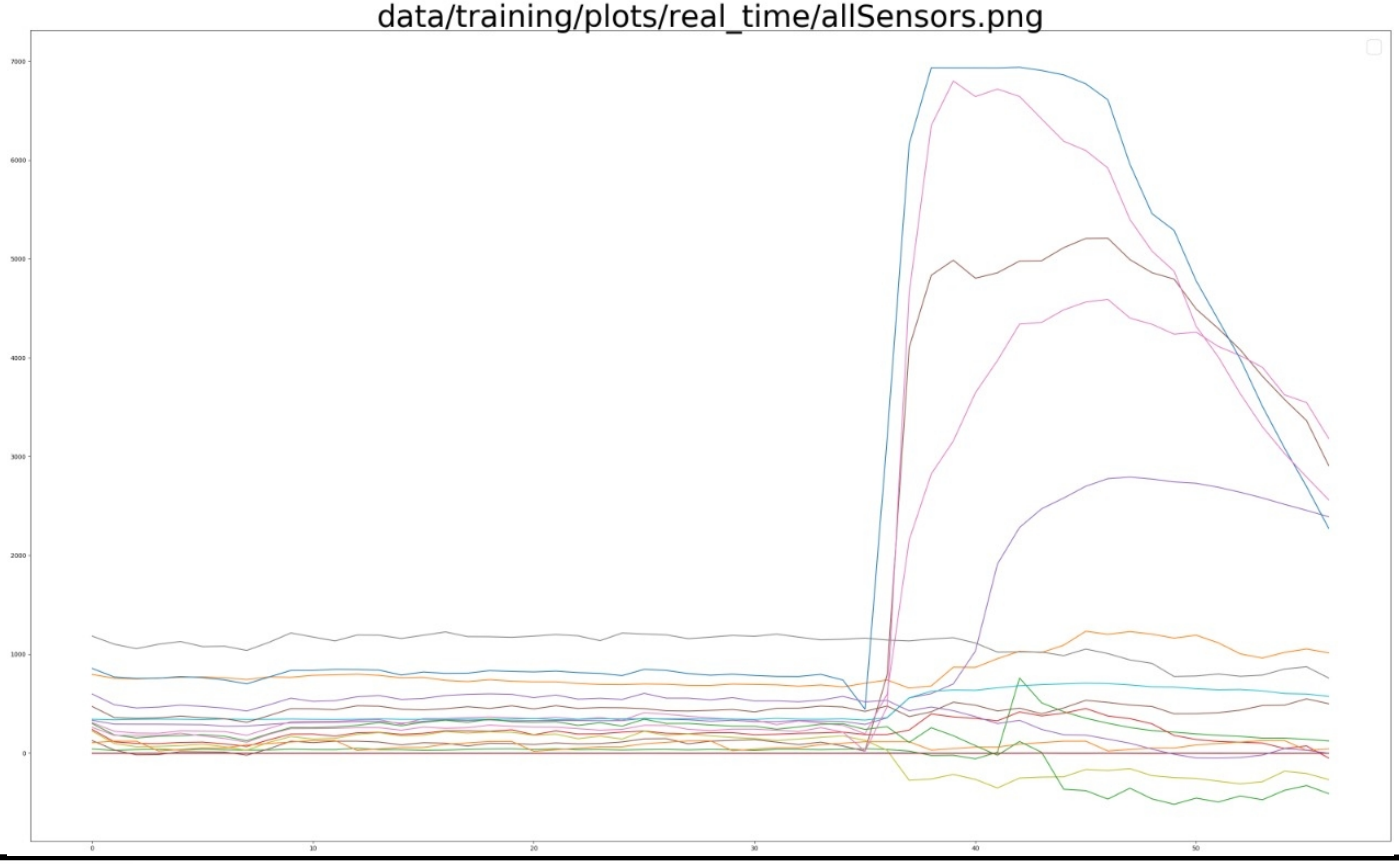
- Data acquisition
  - Decryption delay

```
def handler(self, data):

    self.counter += 1
    if self.counter == 15:
        assert data[0] == 0
        tasks.put_nowait(''.join(map(chr, data[1:])))
        self.packets_received += 1
        self.counter = 0
    return True
```

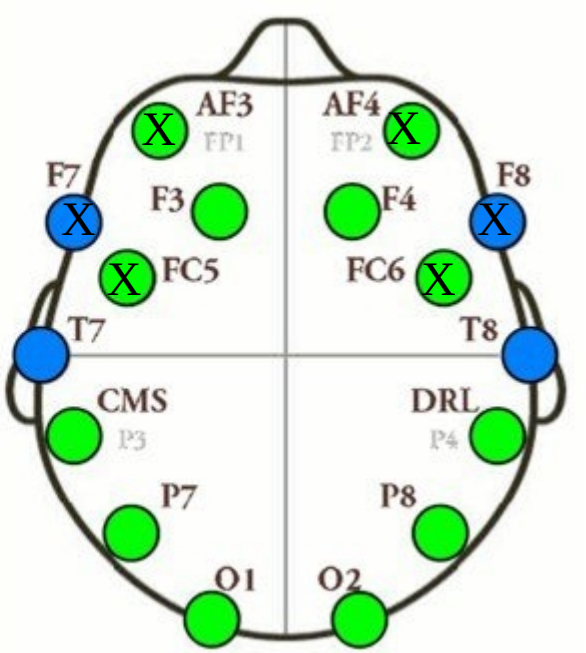
Feature Selection

- Feature reduction from 16 active sensors to 6



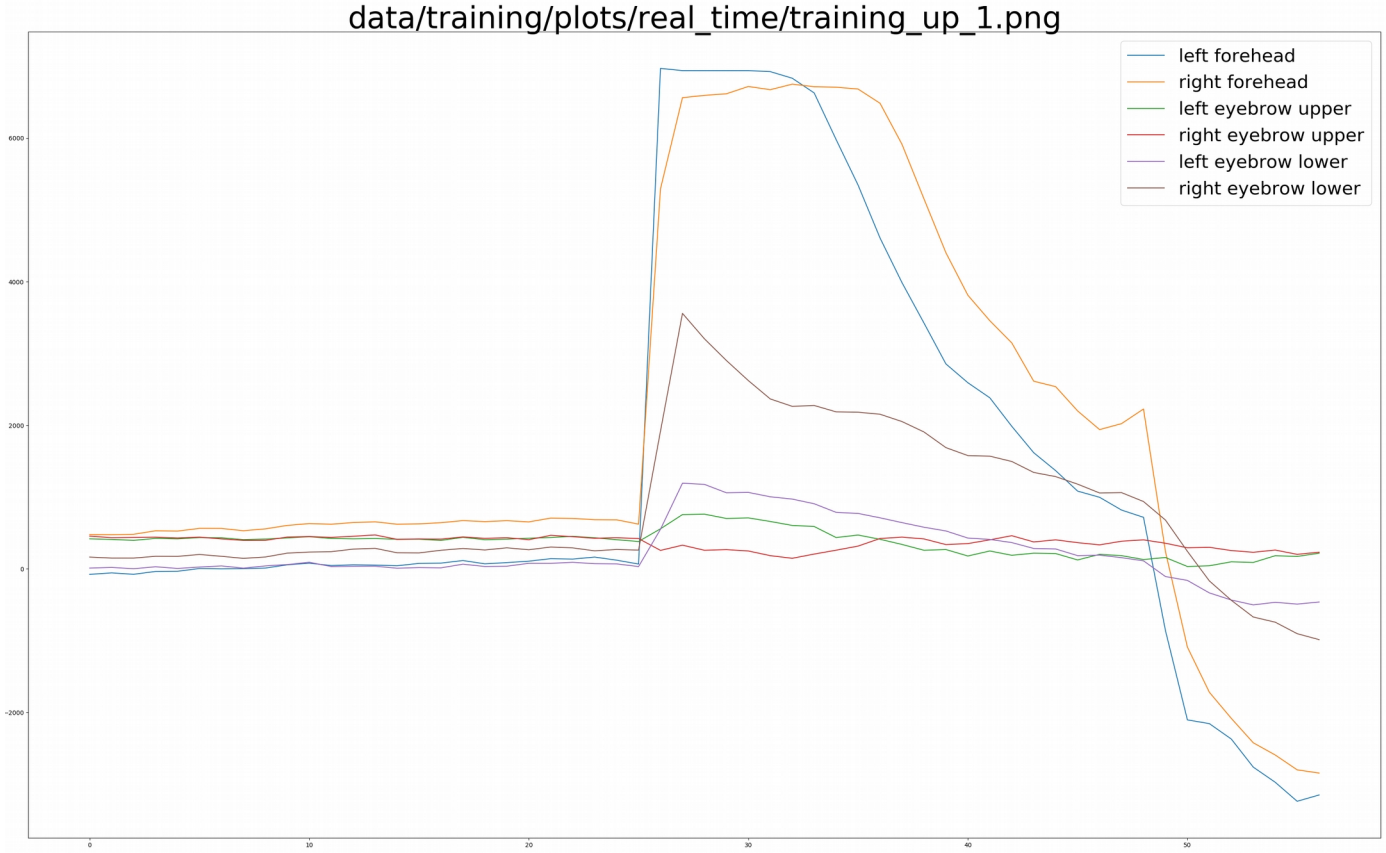
Feature Selection

- Feature reduction from 16 active sensors to 6



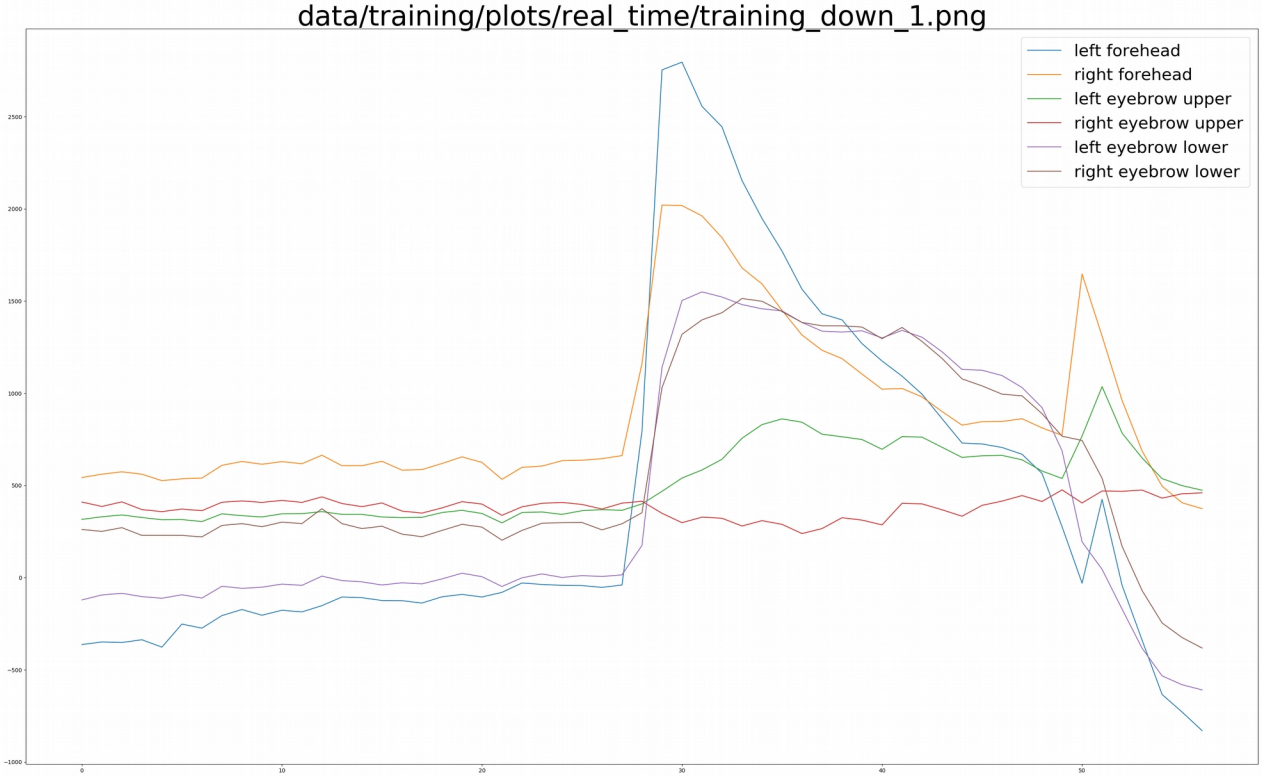
Feature Selection

- Feature reduction from 16 active sensors to 6



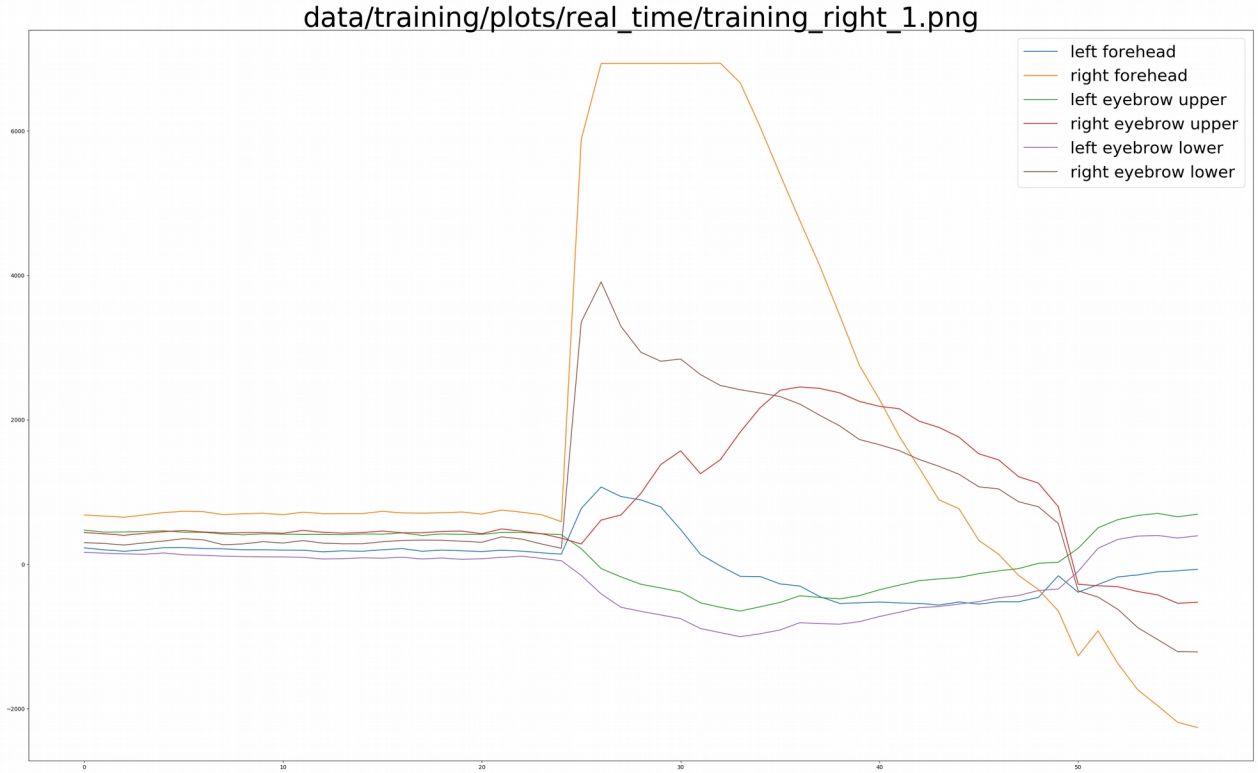
Feature Selection

- Feature reduction from 16 active sensors to 6



Feature Selection

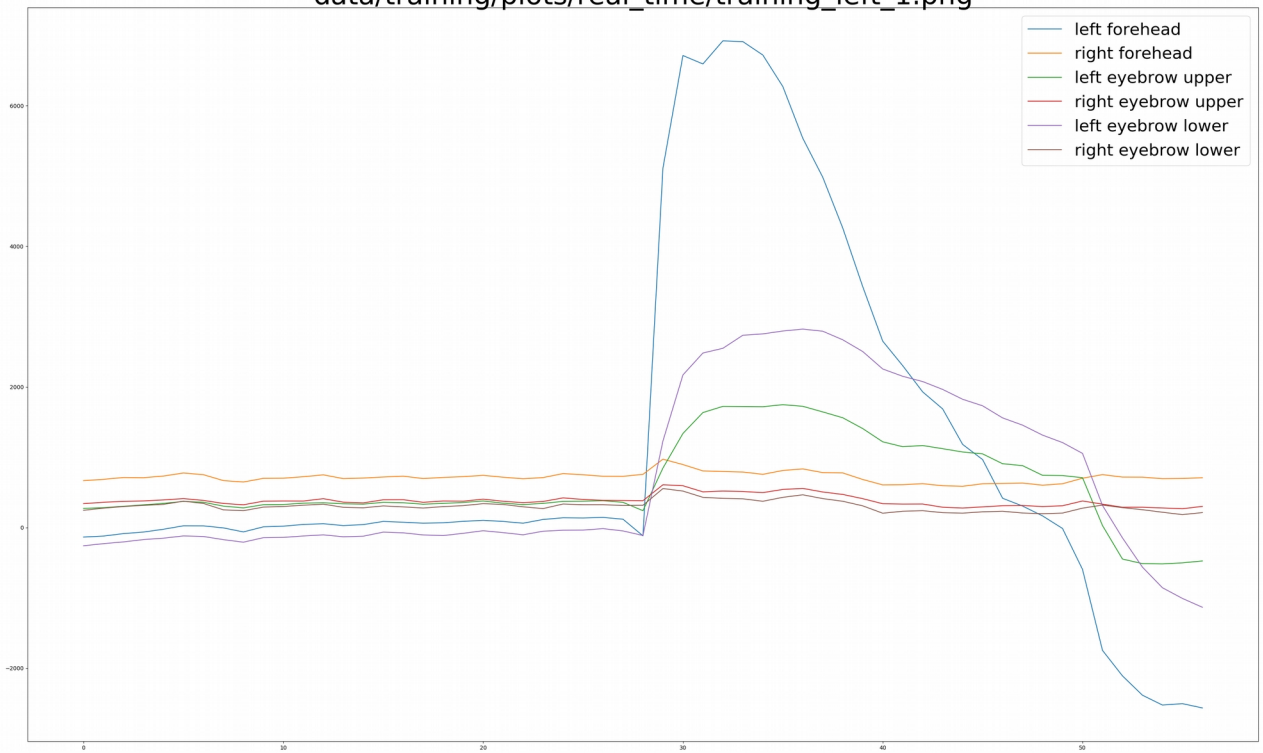
- Feature reduction from 16 active sensors to 6



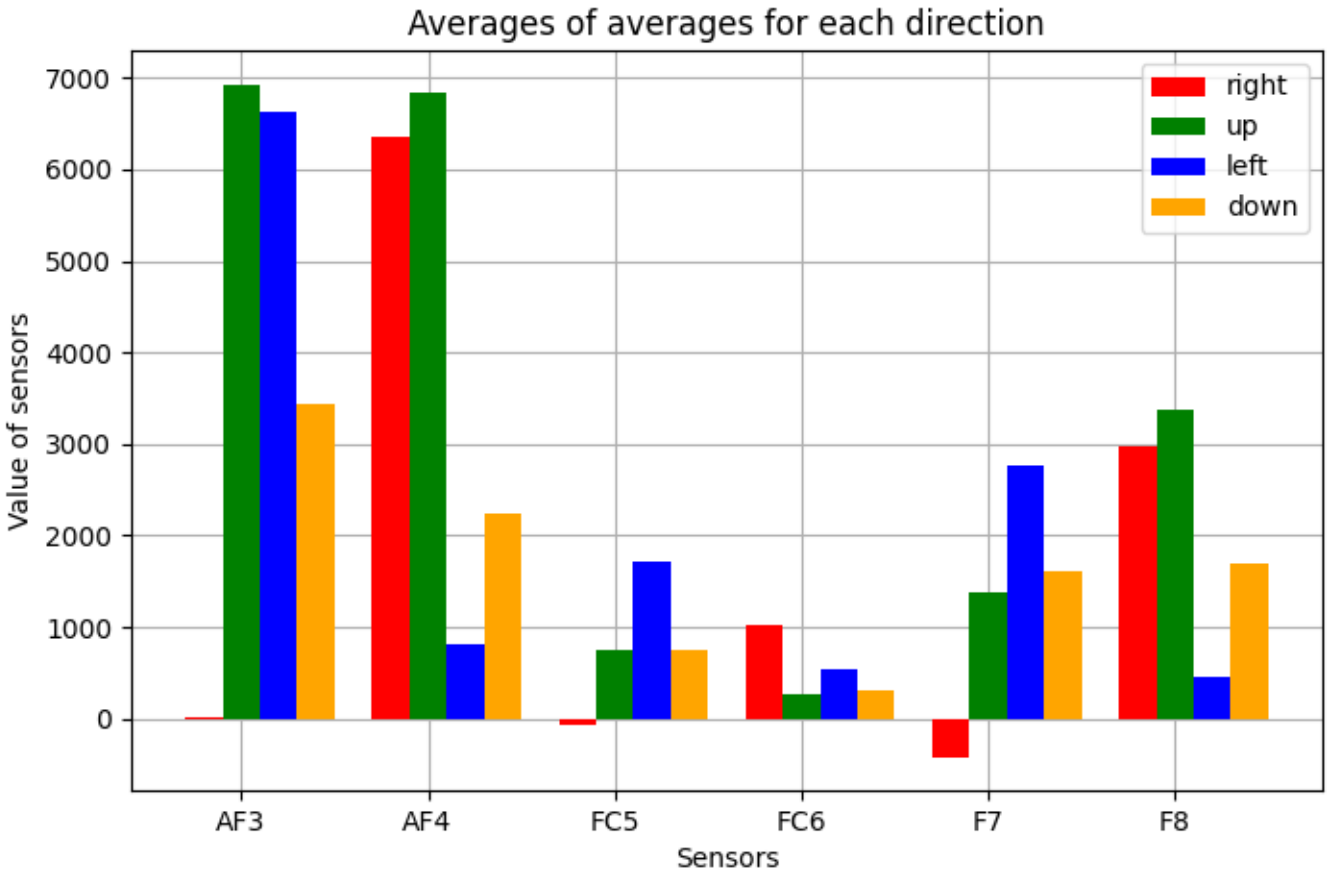
Feature Selection

- Feature reduction from 16 active sensors to 6

data/training/plots/real_time/training_left_1.png



Feature Selection



Spike detection + feature extraction

```
while self.running:
    for k in enumerate(self.sensors):
        if startClassification:
            toClassify = [self.sensors["AF3"]['value'],
                          self.sensors["AF4"]['value'],
                          self.sensors["FC5"]['value'],
                          self.sensors["FC6"]['value'],
                          self.sensors["F7"]['value'],
                          self.sensors["F8"]['value']]
            self.classification(toClassify)
            startClassification = 0

        name = k[1]
        if name == 'AF3':
            if (self.sensors[name]['value'] - prevAf3) > 2500:
                startClassification = 1
            prevAf3 = self.sensors[name]['value']
        elif name == 'AF4':
            if (self.sensors['AF4']['value'] - prevAf4) > 2500:
                startClassification = 1
            prevAf4 = self.sensors[name]['value']
        elif name != "FC5" or name != "FC6" or name != "F7" or name != "F8":
            continue
```

Data Training

For one movement :

- extract 30 samples x 6 se
  x 5 recordings
- correct the array length
- save the array in npy file

```
for s in sensorDataPair:
    for i in range(len(DESIRED_SENSOR_ARRAY)):
        if s[0] == DESIRED_SENSOR_ARRAY[i]:
            value = int(s[1])
            specificSensorData[i].append(value)
np.save("data/training/" + direction + ".npy", specificSensorData)
masterSensorArray.append(specificSensorData)
```

```
for j in range(4):
    data = np.load(filename + str(j) + '.npy')

    af3 = data[0]
    af4 = data[1]

    prev3 = af3[0]
    prev4 = af4[0]
    for i in range(len(af3)):
        if ((af3[i]-prev3) > 1000) or ((af4[i]-prev4) > 1000) :
            cutoff = i
            break
        prev3 = af3[i]
        prev4 = af4[i]

    for d in range(len(DESIRED_SENSOR_ARRAY)):
        temp = data[d]
        cutoffArray[d] = temp[cutoff+1:cutoff+6]

    np.save(filename + "cutoff_" + str(j) + ".npy", cutoffArray)
```

When there is a spike, take the next samples and get an average for each sensor

Data Classification

scipy.spatial.distance.euclidean(*u, v, w=None*)

Computes the Euclidean distance between two 1-D arrays.

The Euclidean distance between 1-D arrays *u* and *v*, is defined as

$$\|u - v\|_2$$

$$\left(\sum \left(w_i |(u_i - v_i)|^2\right)\right)^{1/2}$$

```
minimum = min(distLeft, distRight, distUp, distDown)
if minimum == distUp:
    classification = "up"
elif minimum == distDown:
    classification = "down"
elif minimum == distRight:
    classification = "right"
elif minimum == distLeft:
    classification = "left"
```

Conclusion

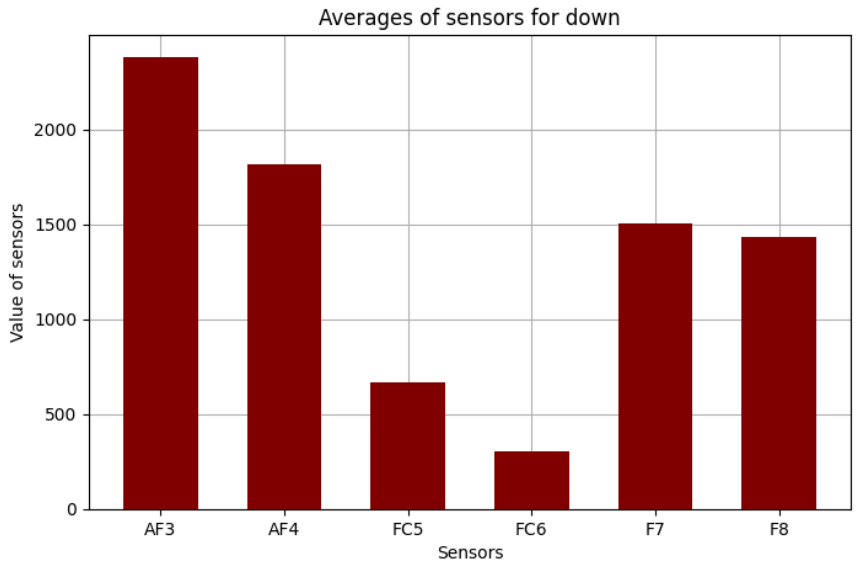
- Successful extraction, training and classification of the real-time data comming from the EPOC headset.
- Accuracy of the classification ~ 90%.

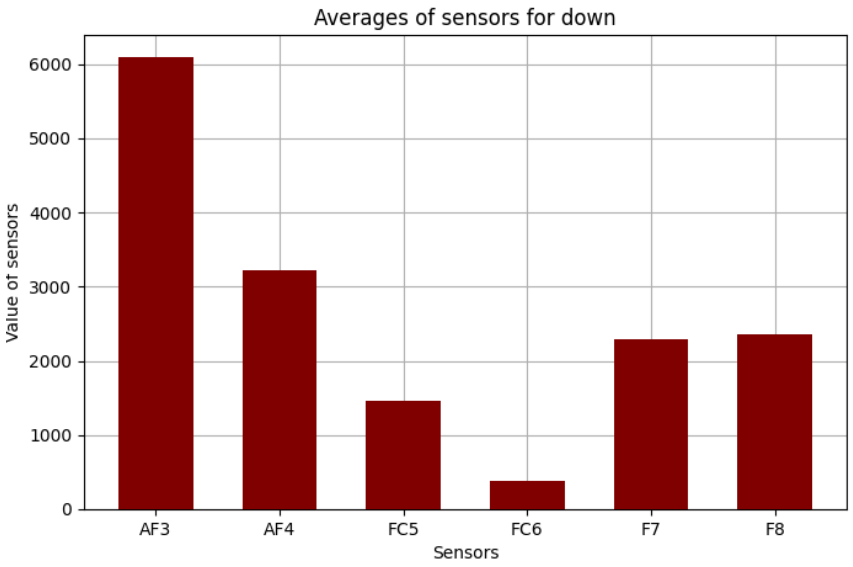
Improvements that can be done

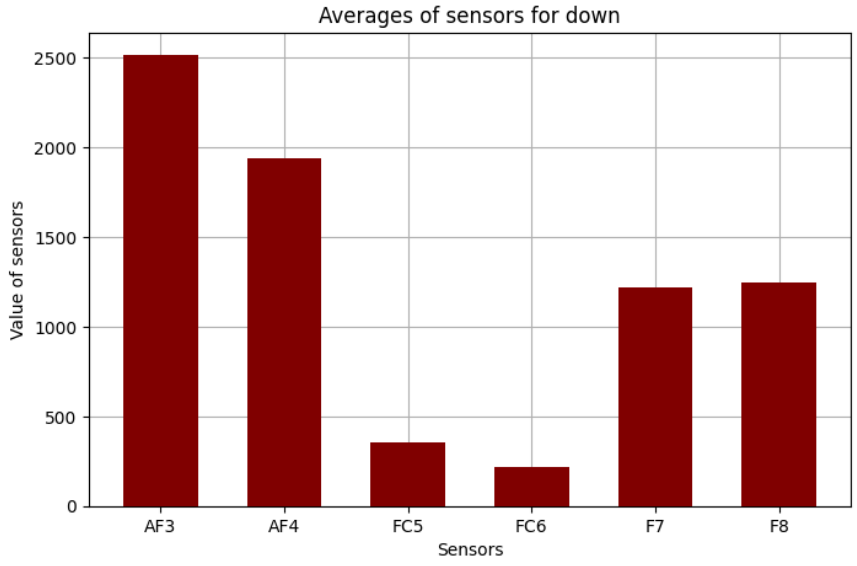
- With the same code with very minor modifications, but with a new set of sensors, we could utilize the real brain electrical activity rather than the muscle movement.

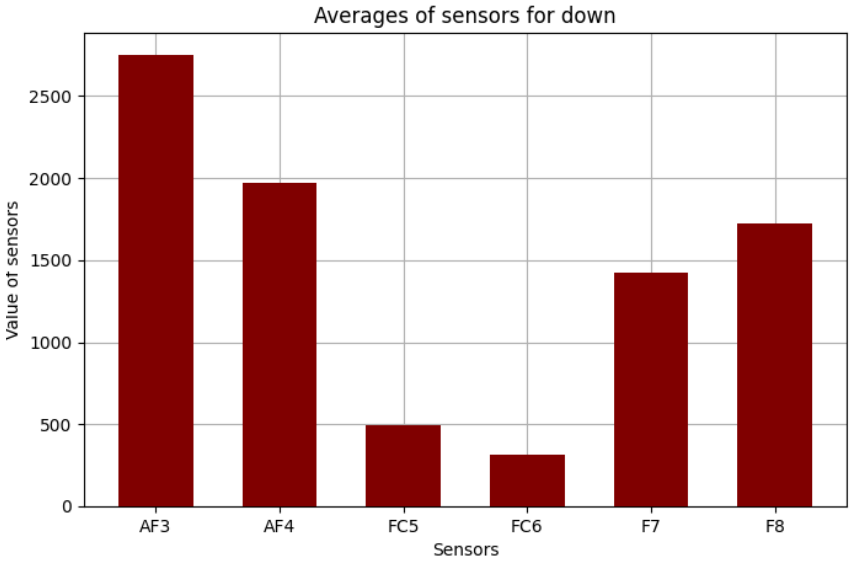
Conclusion

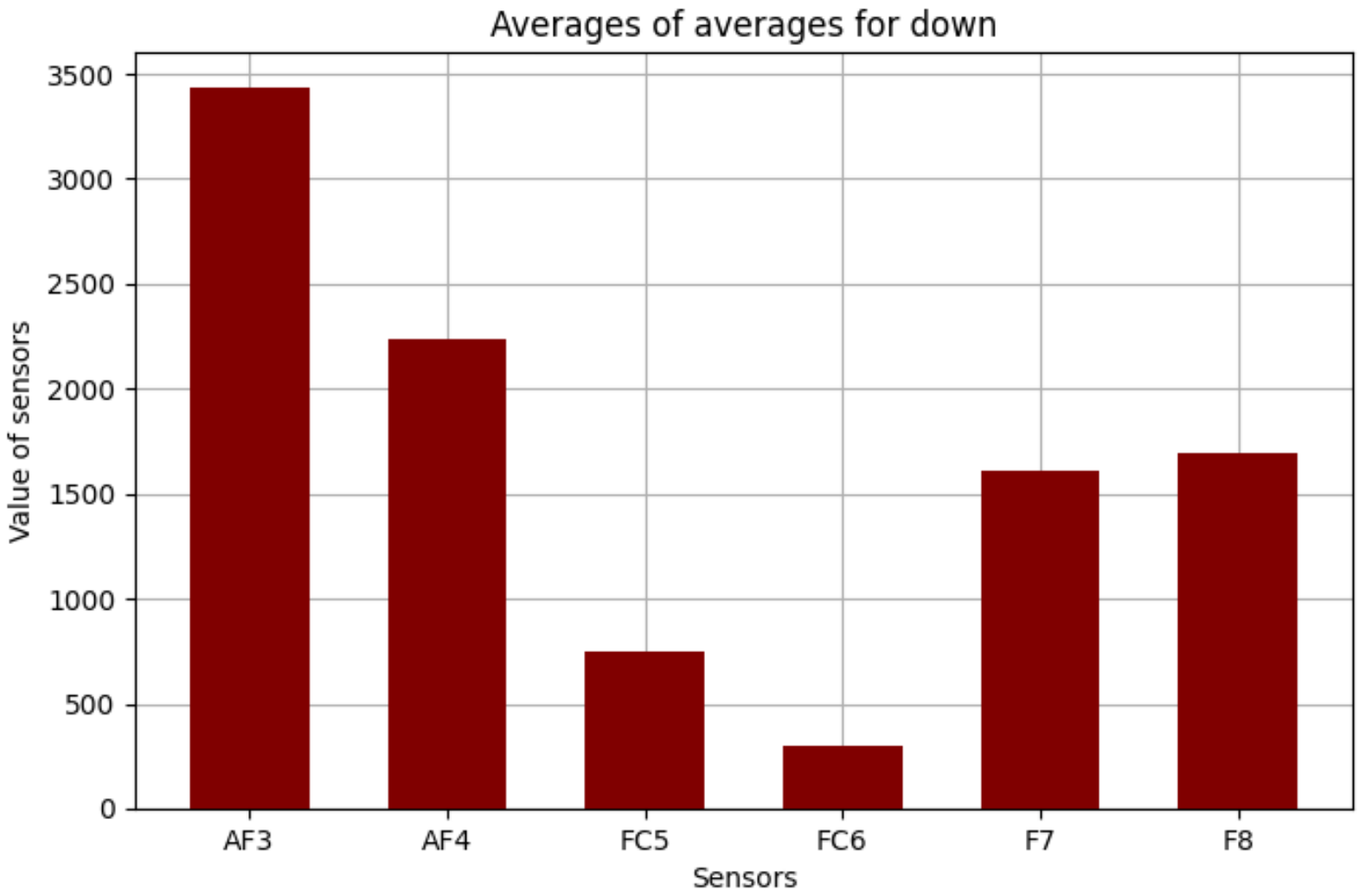
Thank You for your attention!


Averages of sensors for down
Value of sensors
0
500
1000
1500
2000
AF3
AF4
FC5
FC6
F7
F8
Sensors


Averages of sensors for down
Value of sensors
0
1000
2000
3000
4000
5000
6000
AF3
AF4
FC5
FC6
F7
F8
Sensors


Averages of sensors for down
Value of sensors
0
500
1000
1500
2000
2500
AF3
AF4
FC5
FC6
F7
F8
Sensors


Averages of sensors for down
Value of sensors
0
500
1000
1500
2000
2500
AF3
AF4
FC5
FC6
F7
F8
Sensors


Averages of averages for down
Value of sensors
3500
3000
2500
2000
1500
1000
500
0
AF3
AF4
FC5
FC6
F7
F8
Sensors