UNIVERSITY OF TECHNOLOGY, SYDNEY
Faculty of Engineering and Information Technology

Driver Fatigue Detection with EEG

by

Thomas Wengerter

Report Assignment 3
submitted for the Subject

Biomedical Instrumentation

Sydney, Australia

2019

Contents

Chapter 1

Introduction

In 2018, 20% of all fatal road accidents were caused by driver fatigue (TAC 2018). Long drives are common in Australia, yet many drivers do not take the symptoms of fatigue seriously and threaten their and other peoples lives. Within a 4 seconds sleep, a car with a speed of 100 km/h travels 111 meters without a driver in control.

Especially young drivers, truck drivers or shift workers have a higher risk to be involved in crashes due to fatigue. To reduce the number of fatal road accidents caused by driver fatigue, a technology to detect the drivers fatigue is desired. Visual sensors checking facial expressions, yawning and eye closure time are very effective for this task, but object to the visual information which can be erroneous or obstructed by darkness. An alternative is to sense the drivers brainwave activities with Electroencephalography (EEG). An EEG device collects electrical waves with sensors attached across the patient's scalp. In the conventional 10-20 system, the sensor positions are defined and named. The amplitudes of the sensed waves is extremely small and filtering is required to remove noise or power supply interference.

Generally, EEG signals are split into 4 different wavebands described by (Subha et al. 2010). Low frequencies from 1 to 3 Hz are considered as the delta band. Activity in this frequency range only occurs in deep sleep stages or with infants. Likewise, activity in the theta band from 3 to 8 Hz is measured with children or in a state of strong drowsiness and also while sleeping. Alpha waves range from 8 to 12 Hz and are particularly present in the posterior head and in active head regions. They generally occur in relaxed states of adults. The highest frequency

band contains the beta waves from 12 to 32 Hz. Beta waves are dominant in the frontal regions and are associated with alert or scared emotions.

Many previous publications focus on the effects of fatigue on the brain activities Hu and Min (2018); Subha et al. (2010); Li et al. (2019); Wang et al. (2019); Craig et al. (2012). The deviations between the individual findings emphasize how complex the nature of brain waves actually is. However, some trends can be found in accordance to all the papers reviewed. If a patient fatigues, the alpha and beta wave activities increase generally. Comparing the publications, characteristic effects in the theta and delta band are ambiguous. Also, preferably the EEG channels in the back of the scalp like PO3, POz, PO4, O1h, Oz, O2h gave the best results according to (Wang et al. 2019). (Hu and Min 2018) stated the fatigue detection could also be accomplished with channel TP7 only.

In this report, a classifying method is proposed and implemented. Labeled training EEG data of drivers published by (Min et al. 2017) is used to extract features and train the classifier. Finally, the classification performance of a logistic regression machine learning classifier, a multi-layer neural network and a gradient boosted decision tree classifier are compared with the extracted features PSD, mean, standard deviation, sample entropy and WPT coefficients selected with fuzzy entropy.

Chapter 2

EEG Signal Preprocessing

To develop the desired classifier, realistic EEG data of a driving person has to be collected. Since the setup and the conduction of a data collection in a simulated environment exceeds the scope of the project, the dataset published in (Min et al. 2017) is the base of the training data for the classifier and can be downloaded online. Hu and Min set up a driving simulation task for the patients, where they will be exposed to a monotonous driving task for at least 2 hours while a 40-channel Neuroscan EEG system records the brainwave activities on the patient's scalp with a sampling frequency of $f_\mathrm{s} = 1000$ Hz. The setup of this driving simulator is shown in Fig. 2.1. In total, the available dataset contains of the EEG data of 12 patients. For each patient, one file contains 300 seconds of EEG signal collected while the patient is performing the driving simulation task in the awake state. A second file of 300 seconds EEG data is recorded at the end of the driving simulation task when the driver shows clear symptoms of fatigue like expanded eye closure time, yawning and slower reaction time.

For the raw EEG sensor signals, multiple preprocessing steps are necessary to reduce interference or remove irrelevant signal components from the EEG signal. First, the .cnt files from (Min et al. 2017) are imported to the signal processing software MATLAB. After an visual inspection of the published data, the signal processing is implemented in a MATLAB script. A snippet of an unfiltered, original EEG sensor signal sensed on channel TP7 is shown in Fig. 2.2. First, the EEG signals are high-pass filtered at a frequency of 0.5 Hz to remove any DC components. A

Figure 2.1 : Setup of the driving simulation (Hu and Min 2018).

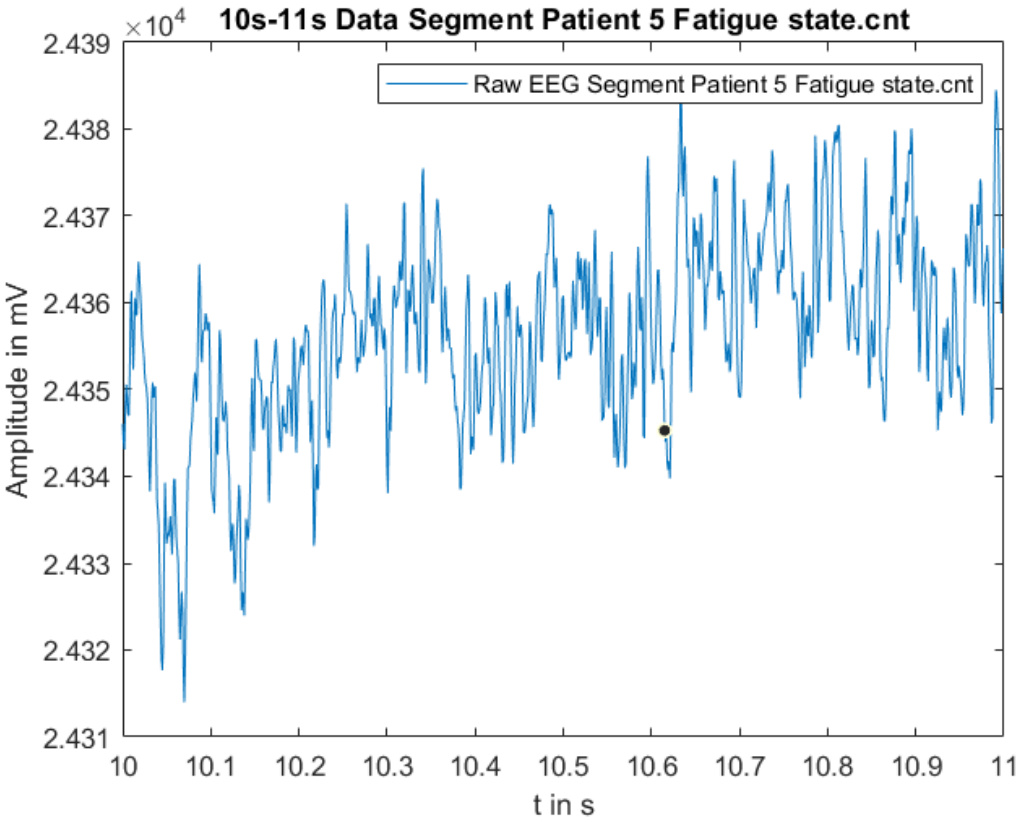

Figure 2.2 : Segment of a unfiltered EEG sensor signal in MATLAB.

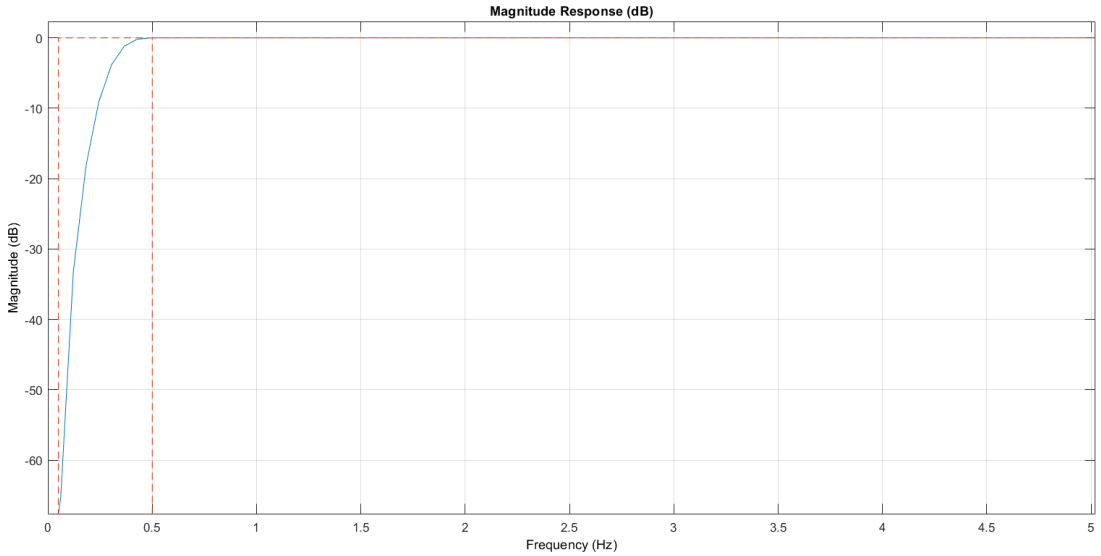

Figure 2.3 : Frequency response of the high-pass filter in MATLAB.

FIR kaiser window filter is implemented in MATLAB with the settings in Listing 2.1 resulting in a frequency response shown in Fig. 2.3.

```
1  hpf = designfilt('highpassfir', 'StopbandFrequency',
       0.1*0.5, 'PassbandFrequency', 0.5, '
       StopbandAttenuation', 80, 'PassbandRipple', 0.01, '
       SampleRate', 1000, 'DesignMethod', 'kaiserwin');
2  fvtool(hpf) % plot freq response
3  EEG{patient,meas} = filtfilt(hpf, EEG{patient,meas}); %
       apply HPF
```

Listing 2.1: High-pass filter design in MATLAB.

MATLAB's 'filtfilt()' function applies the filter first in forward and afterwards reverses the filtered signal and runs it through the filter again. This zero-phase filter ensures that no time shift is added to the signal and every event of the original signal stays at the exact same time position in the filtered signal.

Next, the low-pass filter attenuates the high frequency noise of the EEG signals. Since the relevant frequency band for brainwave activities lies in the range from 1 Hz up to 32 Hz (see chapter 1), the passband of the low-pass filter is designed up

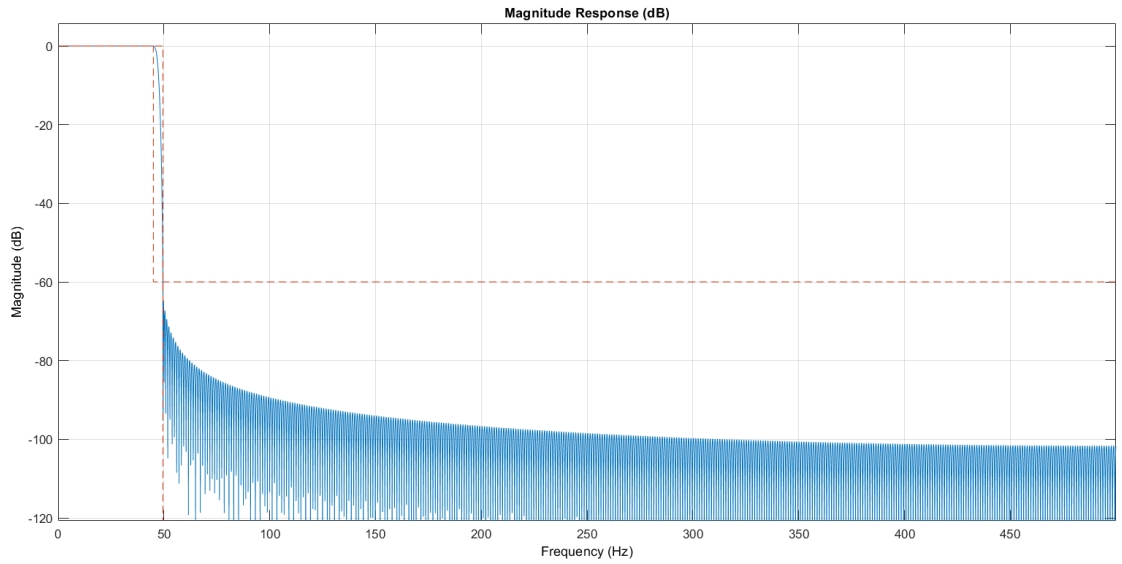

Figure 2.4 : Frequency response of the low-pass filter in MATLAB.

to 45 Hz. Again, the FIR filter used follows a kaiser window design with settings as shown in Listing 2.2. This selection also filters the interference of the power supply located at 50 Hz dominating the unfiltered signal (Hu and Min 2018). The low-pass filter's frequency response is plotted in Fig. 2.4.

```
1  lpf = designfilt('lowpassfir', 'PassbandFrequency', 45,
       'StopbandFrequency', 45*1.1, 'PassbandRipple', 0.01,
       'StopbandAttenuation', 60, 'SampleRate', 1000,'
       DesignMethod','kaiserwin');
2  fvtool(lpf) % plot freq response
3  EEG{patient,meas} = filtfilt(lpf,EEGraw{patient,meas});
       % apply LPF
```

Listing 2.2: Low-pass filter design in MATLAB.

The filtered signal is split into time segments of arbitrary length to create a larger amount of separate signal segments from the 300 seconds long EEG record. Thus, a set of 600 single 1 second signal segments (300 normal, 300 fatigue) are obtained from each patient in each channel and labelled with '0' for normal and '1' for fatigue. Thus, the total amount of the 40-channel EEG 1 second snippets available

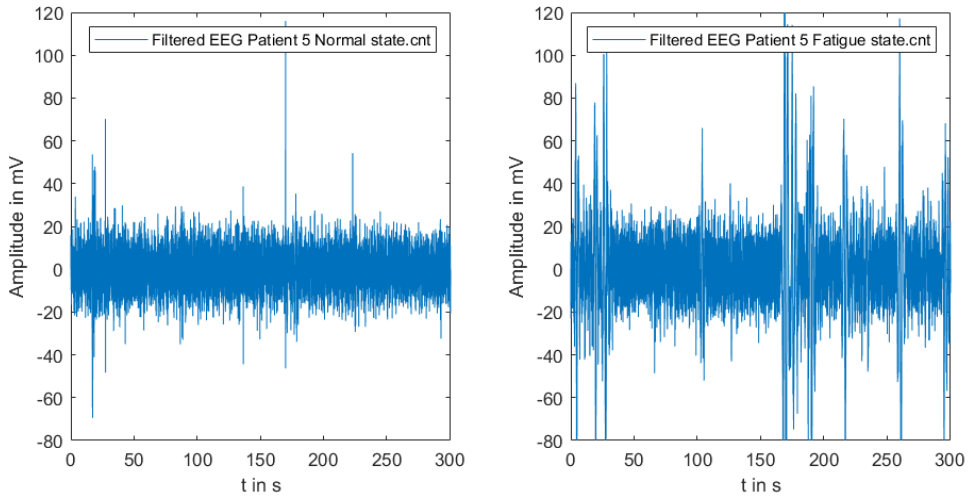

Figure 2.5 : Comparison of the filtered EEG signals on channel TP7.

for classification is $600 * 12 = 7200$.

Fig.2.5 shows the plot of the EEG signal for the same patient in normal and fatigue state. Some differences already visible by inspection. The amplitudes of fatigue EEG is slightly higher than the amplitude of the normal EEG, indicating increased brainwave activities within the passband from 0.5 Hz and 45 Hz. Large peaks in the fatigue signal indicate that the artifacts from blinking and muscular movements become more distinct in the EEG signal when the driver fatigues.

The function 'CNTpreprocess.m' loads the raw EEG sensor files in .cnt format into MATLAB and performs the preprocessing which includes the described steps of filtering and time segmentation. Also, it has the options to produce a number of plots which can be turned on and off by the user. Both the raw EEG signal and the filtered signal can be plotted in time domain or in spectral domain to compare the patient's normal and fatigue state. For each channel, all 7200 labelled time snippets are collected in a MATLAB struct called 'Train' and saved as file 'TrainingDataCh'.

Chapter 3

Feature Extraction

The filtered time domain EEG comprises a large amount of data points (1000 samples) containing only little information for the classifier. Analyzing time domain signals requires complex classifier architectures with sliding windows and pattern recognition similar to classifier used in image recognition or other visual applications. Nevertheless, this type of classifier can hardly extract information about the signal's frequency spectrum and separate certain brainwave activities according to their frequency band theta, gamma, alpha or beta. To create a simpler, lower dimensional classifier input vector with more concentrated information about the fatigue, the characteristics of the filtered EEG need to be summarized by calculated signal parameters called features. In (Li et al. 2019) Chen stated that the most significant changes between normal and fatigue state are observable in the alpha and frontal mid-central beta band brainwave activities. This is also illustrated in Fig. 3.2, where red coloured areas on the scalp mark the location of strong brainwave activities measured in the corresponding wave band. Therefore, the feature extraction focuses on analyzing particularly the alpha and beta wave activities. This has also the advantage that an independent component analysis (ICA) becomes obsolete. ICA is commonly used in EEG analysis to remove artifacts from eye closure or muscular movements because it separates individual brain event signals with different sources (Onton and Makeig 2006). Both appear as large peaks in the EEG spectrum and are misleading when extracting features from the signal as shown in the plot in Fig.3.1. Most of the sensed activity due to artifacts is generally contained in the lower frequency regions gamma and theta (1-8 Hz). Therefore, artifact

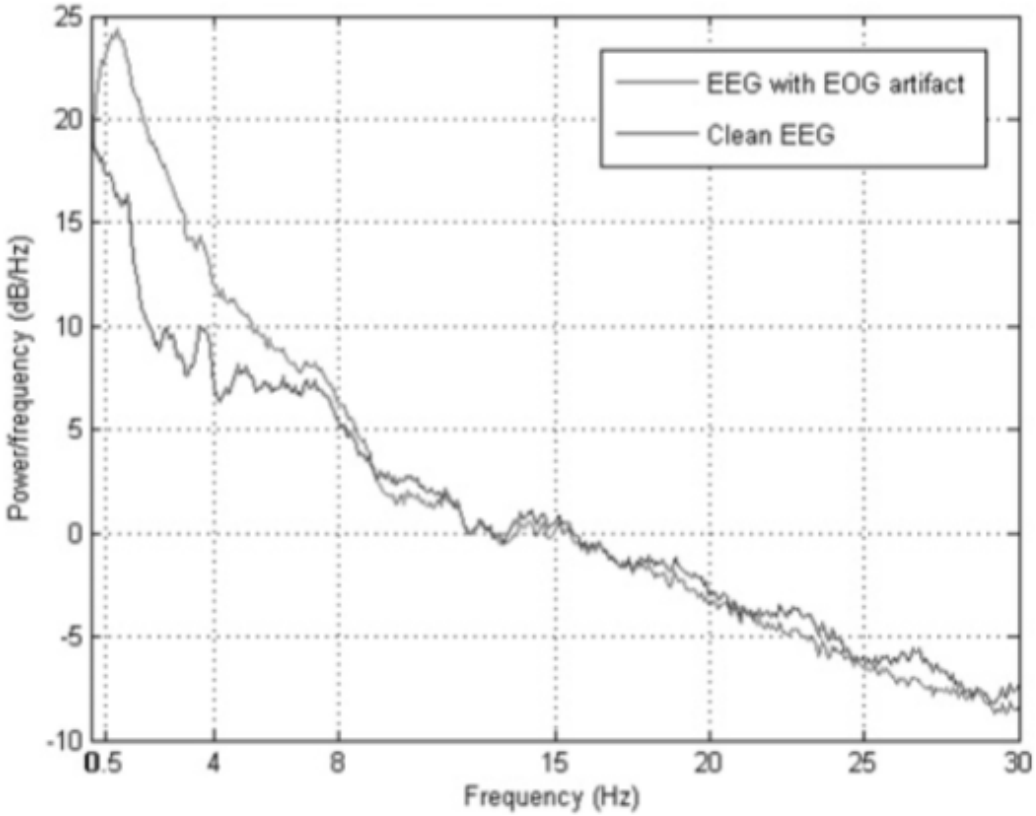

Figure 3.1 : Effects of artifacts on the EEG spectrum.

removal with independent component analysis can be omitted when extracting the features primarily from the higher frequency bands alpha and beta (Li et al. 2019).

To perform the feature extraction running on each of the 1 second 40-channel EEG segments, the second MATLAB script called 'FeatureExtraction.m' is used. It loads the output files from the 'CNTpreprocess.m' function and extracts the individual features for every single time snippet of a channel.

Applying the fast fourier transform (FFT) reveals the activity of different frequencies in the given 1 second time signal. The frequency spectrum S[f] of a discrete

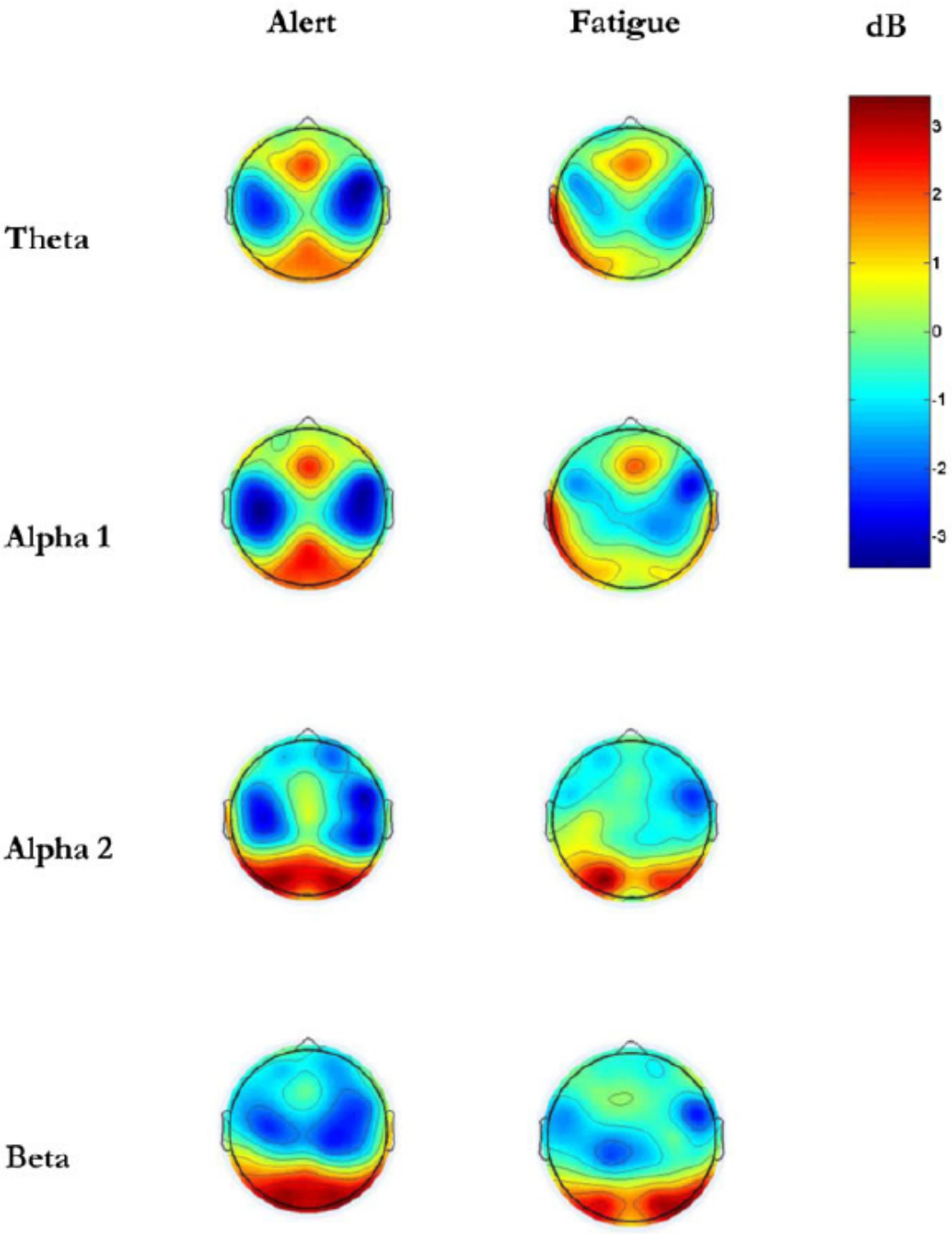

Figure 3.2 : Brainwave activities marked on the scalp, measured in different frequency bands (Craig et al. 2012).

time signal s[n] is generally calculated from

$$S[f] = \sum_{n=-\infty}^{\infty} s[n]\, e^{-j2\pi n} \tag{3.1}$$

with $n = 1, 2, ..., N$ and $f = -\frac{f_s}{2} : \frac{f_s}{N} : \frac{f_s}{2}$. Thus, the activities within the frequency bands alpha and beta can be detected. The alpha and beta band is split in half and the corresponding power spectral density (PSD) inside the subbands is calculated to obtain more precise information about the frequency of the measured brainwave activity. The PSDs represent the first four features and are extracted from the FFT spectrum S[f] by calculating the power spectrum density (PSD) over the relevant frequency bands alpha

$$PSD_{\alpha 1} = \frac{1}{f_s N_{\alpha 1}} \sum_{f \epsilon [8,10] Hz} |S[f]|^2, \tag{3.2}$$

$$PSD_{\alpha 2} = \frac{1}{f_s N_{\alpha 2}} \sum_{f \epsilon [10,12] Hz} |S[f]|^2 \tag{3.3}$$

and beta

$$PSD_{\beta 1} = \frac{1}{f_s N_{\beta 1}} \sum_{f \epsilon [12,17] Hz} |S[f]|^2, \tag{3.4}$$

$$PSD_{\beta 2} = \frac{1}{f_s N_{\beta 2}} \sum_{f \epsilon [17,32] Hz} |S[f]|^2, \tag{3.5}$$

where N_α and N_β are the total number of frequency samples $\frac{f_s}{N}$ within the frequency bands alpha and beta. The corresponding MATLAB code is presented in Listing 3.1.

```
1  % PSD of FFT segments alpha1, alpha2, beta1, beta2
2  Alpha = Train.dataFD(:, alpha(1):alpha(2)); %extract
       window of alpha waves
3  Beta = Train.dataFD(:, beta(1):beta(2)); %extract window
       of beta waves
4  %alpha1 [8 10]       alpha2 [11 13]
5  PSDalpha1 = 10*log(sum(1/(fs*size(Alpha(:,1:floor(size(
       Alpha,2)/2)),2)).*abs(Alpha(:,1:floor(size(Alpha,2)
       /2))).^2,2));
6  PSDalpha2 = 10*log(sum(1/(fs*size(Alpha(:,floor(size(
       Alpha,2)/2)+1:end),2)).*abs(Alpha(:,floor(size(Alpha
       ,2)/2)+1:end)).^2,2));
```

```
7     % beta1 [13 21]      beta2 [22 30]
8     PSDbeta1 = 10*log(sum(1/(fs*size(Beta(:,1:floor(size(
          Beta,2)/2)),2)).*abs(Beta(:,1:floor(size(Beta,2)/2)))
          .^2,2));
9     PSDbeta2 = 10*log(sum(1/(fs*size(Beta(:,floor(size(Beta
          ,2)/2)+1:end),2)).*abs(Beta(:,floor(size(Beta,2)/2)
          +1:end)).^2,2));
```

Listing 3.1: Calculate PSDs in MATLAB.

Next, the mean and the variance of the alpha and beta components in the time signal is calculated. To extract the alpha and beta frequency band from the EEG signal with bandwidth from 0.5 Hz up to 45 Hz, a continuous wavelet transform CWT is applied to the signal. The CWT calculates the convolution of a short oscillating signal with high frequency components, the wavelet, with the time domain signal. Generally, wavelet transforms resolve both time and frequency domain of a time varying signal while the FFT only provides static information about the frequency distribution of the processed sequence (Combes et al. 1990). With this time-frequency resolution, reconstruction of the signal is possible maintaining the correct time sequence of events in the EEG signal. MATLAB provides frequency selective inverse CWT, which is comparable to band-pass filtering the signal for the selected frequency range as shown in Listing 3.2. Thus, alpha and beta signal can be extracted from the EEG data.

```
1     [wt,freq] = cwt(Train.data(set,:),fs);
2     alphaTD(set,:) = icwt(wt,freq,alpha,'SignalMean',mean(
          Train.data(set)));
3     betaTD(set,:) = icwt(wt,freq,beta,'SignalMean',mean(
          Train.data(set)));
```

Listing 3.2: Extract alpha and beta signal with inverse CWT in MATLAB.

From these filtered signals, the mean of the alpha and beta signal is calculated with

$$\mu_\alpha = \frac{1}{N_\alpha} \sum_{n=1}^{N_\alpha} s_\alpha[n] \tag{3.6}$$

$$\mu_\beta = \frac{1}{N_\beta} \sum_{n=1}^{N_\beta} s_\beta[n] \tag{3.7}$$

and added to the features. Also, the variances of the alpha and beta signals

$$\sigma_\alpha = \sqrt{\frac{1}{N_\alpha - 1} \sum_{n=1}^{N_\alpha} |s_\alpha[n] - \mu_\alpha|^2} \tag{3.8}$$

$$\sigma_\beta = \sqrt{\frac{1}{N_\beta - 1} \sum_{n=1}^{N_\beta} |s_\beta[n] - \mu_\beta|^2} \tag{3.9}$$

are stated as next features of the signal.

Recent publications use various entropies to describe the signal characteristics. Entropies origin from the information theory where the Shannon entropy describes the general complexity of a signal or random variable X

$$H(X) = \sum_{x \epsilon X} -p(x) \log_2 p(x)\,. \tag{3.10}$$

The result of the Shannon Entropy estimates the amount of binary bits to encode X.

For the EEG analysis, the sample entropy is selected as feature by (Tran et al. 2007) and in multiple following papers. It is a useful tool to observe the complexity of a short, noisy time series data sequence by finding similar vectors of various length within the sequence. The detection is constraint with an accuracy factor r (Tran et al. 2007). Sample entropy is generally an improved version of approximate entropy and is calculated with the formula (Wang et al. 2019)

$$SamEn(m, r, N) = -\ln \frac{B^{m+1}(r)}{B^m(r)} \tag{3.11}$$

with

$$B^m(r) = \frac{1}{N - m} \sum_{i=1}^{N-m} C_i^m(r), \tag{3.12}$$

where N is the number of samples contained in the time signal, r represents the noise threshold for similarity and m is the dimensionality of the sub-sequences within the

time signal. For EEG applications, $r = 0.2 * \sigma$ and $m = 2$ is found to give reasonable results. The correlation vector C measures similarities within the time domain signal and is calculated with the two m-dimensional vectors from the time signal $s[n]$

$$C_i^m(r) = \frac{1}{N - m + 1} num\{d|S(i), S(j)| \leq r\}\,. \tag{3.13}$$

Here, $num\{d|S(i), S(j)| \leq r\}$ counts the number of subsequences with starting index j, that match the requirement that the distance $d|S(i), S(j)|$ between two different segments of the time signal is smaller than the similarity threshold r. This distance $d|S(i), S(j)|$ is defined as the maximum difference

$$\max_k |s[i + k] - s[j + k]|. \tag{3.14}$$

A low sample entropy indicates a high self similarity in the signal. Tran stated, that fatigue and stress lead to a reduction of signal complexity which means that the sample entropy increases (Tran et al. 2007). Sample entropy is individually applied to the alpha, beta and the full band signal to extract three feature values.

Spectral entropy is the second entropy type used in the features. It is obtained by applying the conventional Shannon Entropy on the PSD of the signal. Thus, a noisy signal has a high spectral entropy, while a signal containing coded information has a relatively low spectral entropy (Wang et al. 2019).

Additionally, the approach presented by (Khushaba et al. 2011) is implemented to feed the classifier performance with relevant features. Khushaba applied a wavelet packet (WPT) transform to the EEG signal to split the signal into multiple filtered components. The WPT filters the signal iteratively with a high-pass and a low-pass filterbank. Figure 3.3 illustrates the WPT tree, where a filtering step is performed with every branch. Similar to the CWT, the WPT resolves the signal in both time and frequency domain and shows spectral activities for every time step. It is basically splitting up the signals bandwidth into equal frequency bands and represents each

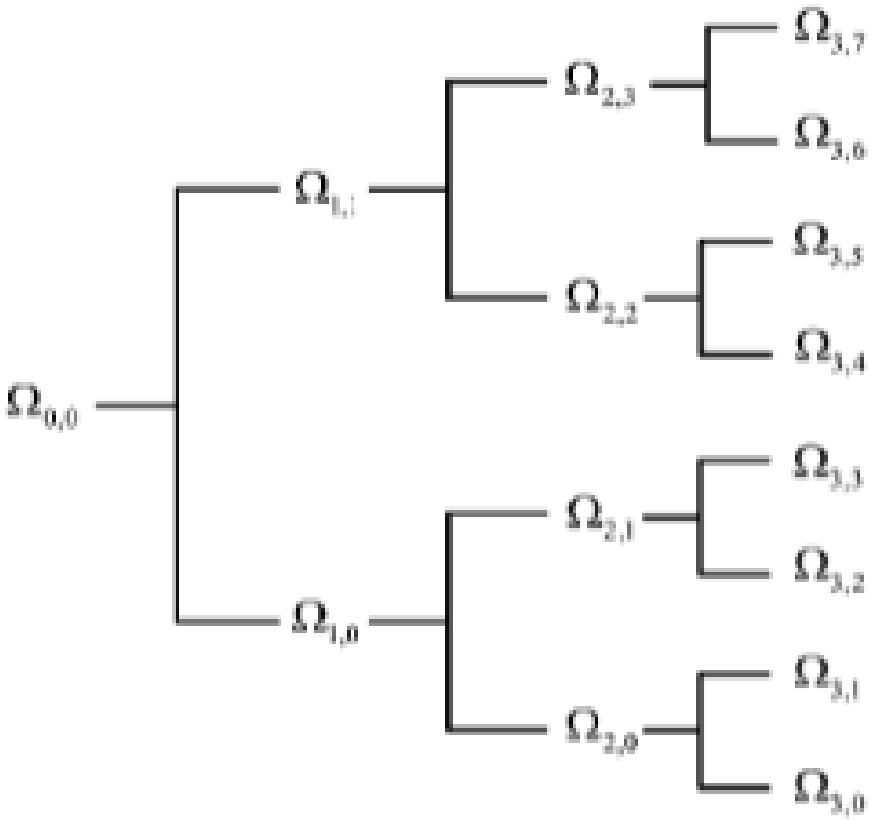

Figure 3.3 : Brainwave activities marked on the scalp, measured in different frequency bands (Craig et al. 2012).

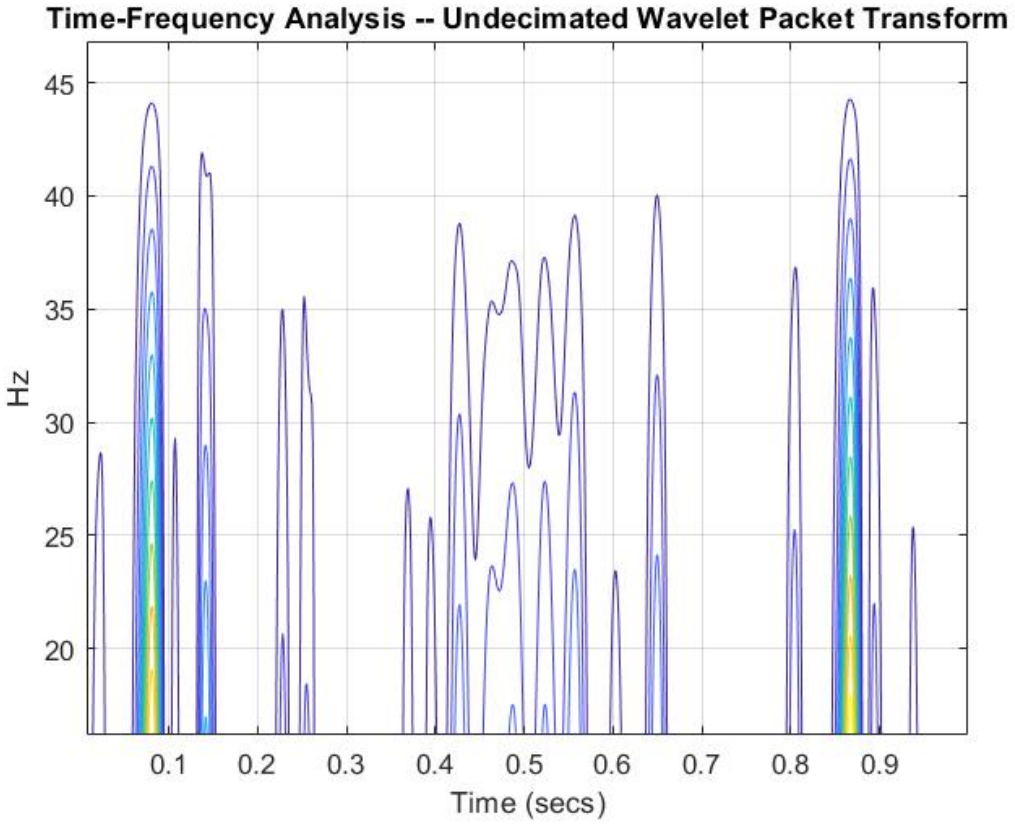

Figure 3.4 : Brainwave activities analyzed with WPT resolving the activities in time and frequency domain.

band in one leaf of a WPT tree level. An illustration of the WPT's time-frequency resolution for an EEG signal is illustrated in Fig. 3.4. For the feature extraction, each decomposition is represented by its normalized filter bank energy (Khushaba et al. 2011) calculated from the sum of the WPT coefficients $w_{j,k,n}$ of the leaf $\Omega_{j,k}$

$$E_{\Omega_{j,k}} = \log(\frac{\sum_n (w_{j,k,n}^T x))^2}{N/2^j}) \tag{3.15}$$

The energies of all WPT leaves for one signal are collected in a vector and the signal's label is added. In Listing 3.3, the WPT and the calculation of the vector $E_{WPT} = [E_{\Omega_{1,1}}, E_{\Omega_{2,1}}, E_{\Omega_{2,2}}, E_{\Omega_{3,1}}, ...]$ is presented.

```
1    %% 5 Khushaba Fuzzy WPT feature extraction
2        wtlvl = 5;
3        trainSets = size(Train.data,1);
```

```
4
5           % 1: Apply WPT to the labeled dataset
6           WIF = {};
7           wtf = zeros(size(Train.data,1),sum(2.^(0:wtlvl-1)));
8           idx = zeros(sum(2.^(0:wtlvl-1)),2);
9           for r = 1:trainSets %size(Train.data,1)
10              wpt = wpdec(Train.data(r,:), wtlvl, 'db4'); %db4
                    wavelet (Wang 2019)
11
12              for lvl = 0:wtlvl-1
13                  for node = 0:2^lvl-1
14                      % E_WPT Feature vector: Energy of each
                            DWT node
15                      WIF{lvl+1,node+1} = log10( 1/(size(Train
                            .data,2)/2^lvl)*sum(wpcoef(wpt, [lvl,
                            node]).^2) );
16                      %collect in feature vector wtf and save
                            leaf indeces in idx
17                      wtf(r,sum(2.^(0:lvl-1))+ node+1) = WIF{
                            lvl+1,node+1};
18                      idx(sum(2.^(0:lvl-1))+ node+1,:) = [lvl,
                            node];
19                  end
20              end
21          end
```

Listing 3.3: Compute the WPT features in vector E_{WPT}.

To reduce the amount of data points in this WPT feature vector E_{WPT}, fuzzy entropy Cao and Lin (2018) is used to find the mutual information (MI)

$$\begin{aligned} I(E_{WPT}, L) &= H(E_{WPT}) + H(L) - H(E_{WPT}, L) \\ &= H(E_{WPT}) - H(E_{WPT}|L) \end{aligned}$$

between the leaf filter bank energies and the corresponding labels L. Based on the normalized MI

$$F_i = I(E_{WPT}(i), label)/H(E_{WPT}(i)), \tag{3.16}$$

the leafs are sorted and the leafs with the highest fuzzy MI are added to the feature set(Khushaba et al. 2011). The added feature is removed from E_{WPT} with all its direct ancestors and descendents. Then, the next highest feature is extracted

and so on. Thus, the vector E_{WPT} is reduced to 21 fuzzy WPT coefficient entries representing characteristics about the most significant WPT components indicating driver fatigue. Listing 3.4 contains the dimensionality reduction with fuzzy entropy.

```
% 2: Fuzzy Entropy and MI calculation of Feature Vectors (rows of wtf)
[I_Cf, I_Cff, I_ff, H_f, H_ff, H_C] = Fuzzy_MI([wtf(1:trainSets,:),Train.labels(1:trainSets)]);
fMI = I_Cf./H_f; %normalized Fuzzy MI

% 3: find WPT nodes with max Fuzzy MI
[fMI,ix] = sort(fMI,'descend');
idxFuzzy = idx(ix,:);

% 4-7: Reduce dimensionality of MI features
% find Features with highest MI I_Cf (most relevant for classification!)
Features = zeros(1,length(fMI));
Fidx = zeros(length(fMI),2);
Qu = fMI;
Quidx = idxFuzzy;

items = 0;
FWPT = {};
while ~isempty(Qu)
    % extract node/feature/column with highest MI
    items = items+1;
    node = Quidx(1,:);
    Features(items) = Qu(1);
    Fidx(items,:) = Quidx(1,:);
    Qu(1) = [];
    Quidx(1,:) = [];
    %check the queue for father or daughter nodes and remove from queue
    rowfather = (Quidx(:,1) == node(1)-1 & Quidx(:,2) == floor(node(2)/2));
    if sum(rowfather)>=1
        %reduce feature dimensionality
        Qu(rowfather)=[];
        Quidx(rowfather,:) = [];
        Features(end) = [];
        Fidx(end,:) = [];
    end
    rowdaughter1 = (Quidx(:,1) == node(1)+1 & Quidx(:,2) == 2*node(2));
```

```
36              rowdaughter2 = (Quidx(:,1) == node(1)+1 & Quidx
                    (:,2) == 2*node(2)+1);
37              if sum(rowdaughter1)>=1
38                  %reduce feature dimensionality
39                  Qu(rowdaughter1)=[];
40                  Quidx(rowdaughter1,:) = [];
41                  rowdaughter2(rowdaughter1) = [];
42                  Features(end) = [];
43                  Fidx(end,:) = [];
44              end
45              if sum(rowdaughter2)>=1
46                  %reduce feature dimensionality
47                  Qu(rowdaughter2)=[];
48                  Quidx(rowdaughter2,:) = [];
49                  Features(end) = [];
50                  Fidx(end,:) = [];
51              end
52
53
54          end
55
56          % Extract FWPT subspaces
57          WTpower = zeros(size(Train.data,1),length(Features))
                ;
58          for feat = 1:length(Features)
59              % find Wavelet Transform Features with highest
                    Fuzzy Entropy
60              % -> wtf(Fidx), find(Fidx,idx)
61              logic = (Fidx(feat,1) == idx(:,1) & Fidx(feat,2)
                    == idx(:,2));
62              WTpower(:,feat) = wtf(:,logic);
63          end
```

Listing 3.4: Dimensionality reduction of the fuzzy WPT feature vector. All selected features are saved in variable WTpower.

This results in a set of in total 32 features $PSD_{\alpha 1}$, $PSD_{\alpha 2}$, $PSD_{\beta 1}$, $PSD_{\beta 2}$, σ_{α}, σ_{β}, μ_{α}, μ_{β}, $SaEn_{\alpha}$, $SaEn_{\beta}$, $FuzzyWPT(1:21)$ and the label to one set of features exported from MATLAB. These features are calculated for all 30 selected relevant channels and added together in one large dataset with in total $N_{\text{channels}} * N_{\text{features}} + label = 30*32+1 = 961$ columns and $N_{\text{epochs}} * N_{\text{states}} * N_{\text{patients}} = 300*2*12 = 7200$ rows, where each row of features plus the label can be used as classifier input. Figure

1	PSDa1	PSDa2	PSDb1	PSDb2	STDa	STDb	meanA	meanB	SampleEntropy	SampleEntropyAlpha	SampleEntropyBeta	Label
2	-142.11	-137.91	-137.7	-137.35	1.9053	3.8133	-0.00037	-0.00037	0.33869	0.2191	0.31458	0
3	-121.14	-132.17	-123	-128.08	1.4485	3.9196	-2.48	-2.48	0.31344	0.28117	0.44493	0
4	-96.702	-105.29	-106.17	-112.79	2.5269	3.6182	6.2108	6.2108	0.24409	0.24651	0.35572	0
5	-95.766	-98.748	-97.294	-107.1	2.8269	4.8656	-12.481	-12.481	0.27968	0.17639	0.42145	0
6	-95.807	-111.6	-99.872	-111.26	2.2101	4.2988	7.1861	7.1861	0.29345	0.20461	0.38851	0
7	-89.528	-91.602	-93.979	-111.29	2.9209	4.2633	-11.593	-11.593	0.30875	0.21819	0.44944	0
8	-96.037	-97.419	-96.015	-106.36	2.5175	4.3479	-6.0647	-6.0647	0.28166	0.21366	0.34436	0
9	-78.576	-80.756	-85.14	-93.822	2.8357	4.1999	-4.0362	-4.0362	0.35205	0.1886	0.35154	0
10	-93.728	-97.505	-97.944	-110.62	2.6301	3.2841	8.7473	8.7473	0.35322	0.19782	0.36767	0
11	-95.841	-99.738	-96.18	-104.19	2.7251	3.6661	-7.8779	-7.8779	0.29939	0.19514	0.41136	0
12	-98.672	-90.893	-93.351	-102.45	3.0456	4.5142	-2.2975	-2.2975	0.32227	0.2209	0.44678	0
13	-79.782	-79.336	-90.59	-103.28	6.3681	5.0118	16.589	16.589	0.28228	0.25818	0.38831	0
14	-86.733	-84.464	-90.317	-97.538	3.5629	3.1429	12.521	12.521	0.39673	0.23227	0.46989	0
15	-87.705	-90.279	-94.195	-102.73	2.97	3.4566	-4.3945	-4.3945	0.29558	0.21025	0.30217	0
16	-93.16	-108.24	-103.71	-114.94	2.612	3.5848	7.2566	7.2566	0.38141	0.16244	0.38194	0
17	-87.341	-95.296	-91.638	-105.44	2.6081	3.8051	0.07631	0.07631	0.39322	0.1505	0.39764	0
18	-95.164	-101.92	-93.633	-108.26	2.141	3.8186	-12.817	-12.817	0.26928	0.24616	0.39774	0
19	-95.498	-101.39	-98.713	-109.91	3.3531	4.5739	0.33398	0.33398	0.2525	0.12363	0.32332	0
20	-95.781	-108.46	-94.767	-102.2	2.4381	4.5735	-4.6211	-4.6211	0.339	0.2316	0.39879	0
21	-96.144	-103.29	-92.055	-105.61	2.4813	4.2767	-4.8261	-4.8261	0.31474	0.16907	0.36644	0

Figure 3.5 : Snippet of the dataset used to train and test the classifiers.

3.5 shows a snippet of the used dataset used. Only the fuzzy WPT coefficients are not shown for clarity in the plot. The dataset is handed over to KNIME to train, test and validate the classifier.

Chapter 4

Classifier

To identify the state of the driver, the extracted EEG features need to be interpreted to detect if the person is awake (desired output '0') or fatigued (desired output '1'). Multiple binary classifier models with supervised learning like support vector machine (SVM), logistic regression machine learning (ML), multiple layer Perceptron/neural network or gradient boosted tree ML are proposed to solve this problem.

SVM is one of the simpler approaches where the ML algorithm tries to split the dataset with lower-dimensional decision boundaries. Common applications are regression or simple classification tasks. Logistic regression ML is considered to be most effective in binary classification tasks. It applies statistical analysis of the dataset to perform the classification. Success rates of the first two approaches was only around 60% accuracy which is definitely not sufficient for the driver fatigue detection. Therefore, it can be assumed that the decision boundaries in feature space are too complex to be modeled by simple SVM or logistic regression ML.

Multi-layer neural networks a more complex model which is capable to resolve highly complex classification problems by splitting up the feature analysis in multiple processing layers. This concept is inspired by the human braincells and is applicable for any kind of supervised learning problem. One layer consists of multiple single perceptrons. A single perceptron, shown in Fig.4.1, is capable to solve a classification task with a linear decision boundary. It takes multiple inputs x_i, weighs them with the corresponding weight w_i and adds a bias b. From the sum of all weighted inputs

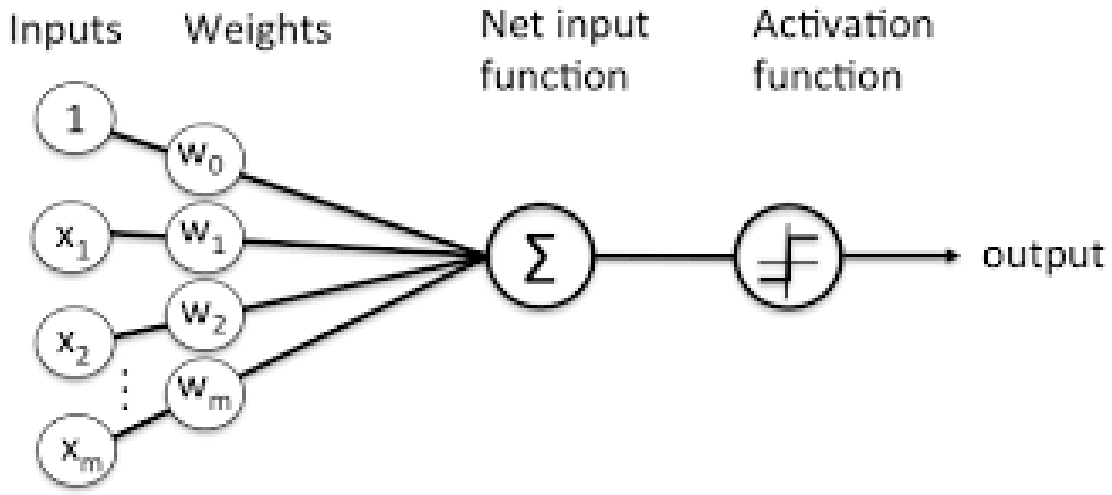

Figure 4.1 : Perceptron model flowdiagram Zahid Hasan (2017).

and the bias results the total input called *net*

$$net = \sum_{i=0}^{N} (w_{\mathrm{i}} x_{\mathrm{i}} + b). \tag{4.1}$$

Evaluating the perceptron's arbitrary differentiable activation function $f(net)$ in the function block gives the output *out*

$$out = f(net)\,. \tag{4.2}$$

Assuming a 2-dimensional labeled input in Fig. 4.2, this model is able to separate the data points with a linear hyperplane by adjusting weights and bias accordingly throughout the epochs in the training process. For this simple example, a linear output function is sufficient. When modelling more complex, non-linear problems, other activation functions like tanH or ReLu functions are popular. Adding multiple perceptrons together to a layer and stacking a set of concentric layers together builds up to the multi-layer neural network as shown in Fig. 4.3. For a binary classifier, the output layer is implemented with a single perceptron applying a sigmoid function to the weighted input of the previous layer net_{out}.

Another approach to solve the EEG classification problem is the gradient boosted

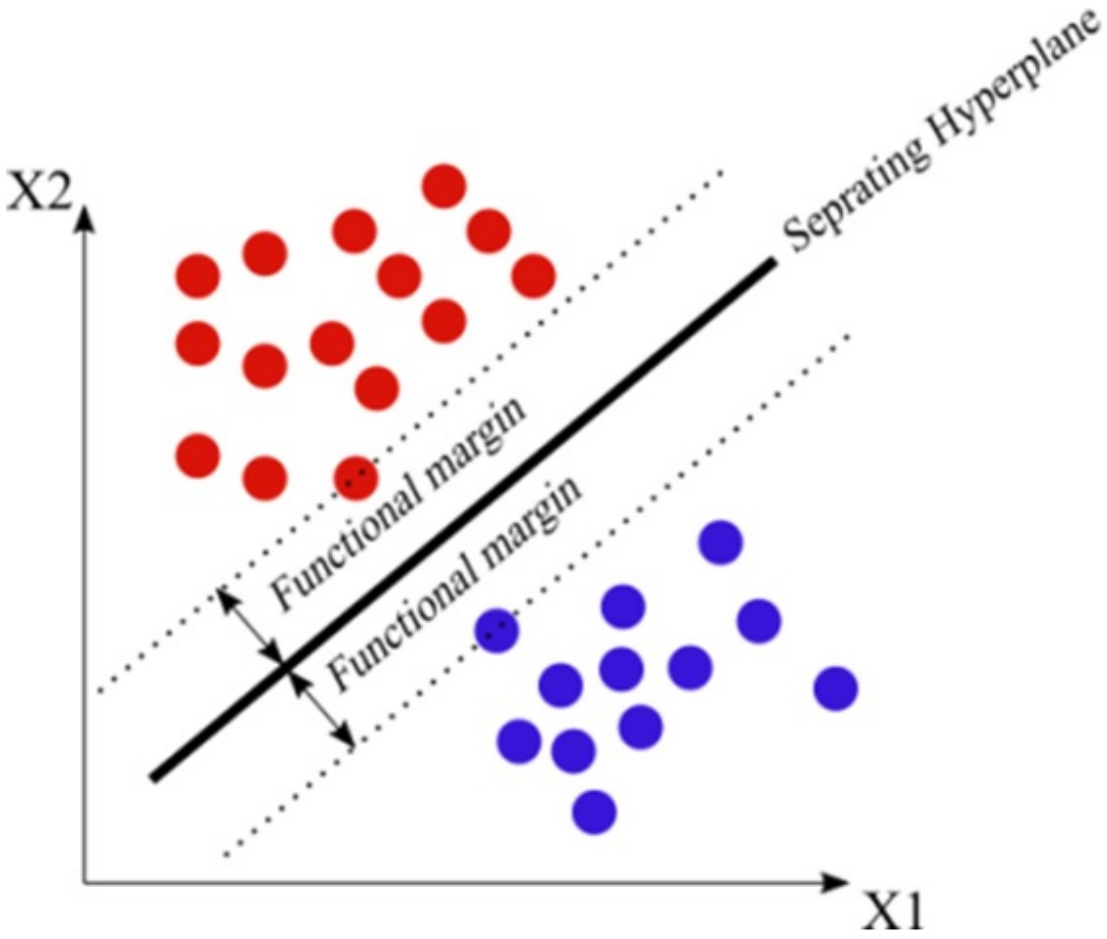

Figure 4.2 : Decision hyperplane for two dimensional perceptron input X1 and X2. All red points are labelled with '1' and blue points with '0' (Chorzepa et al. 2016).

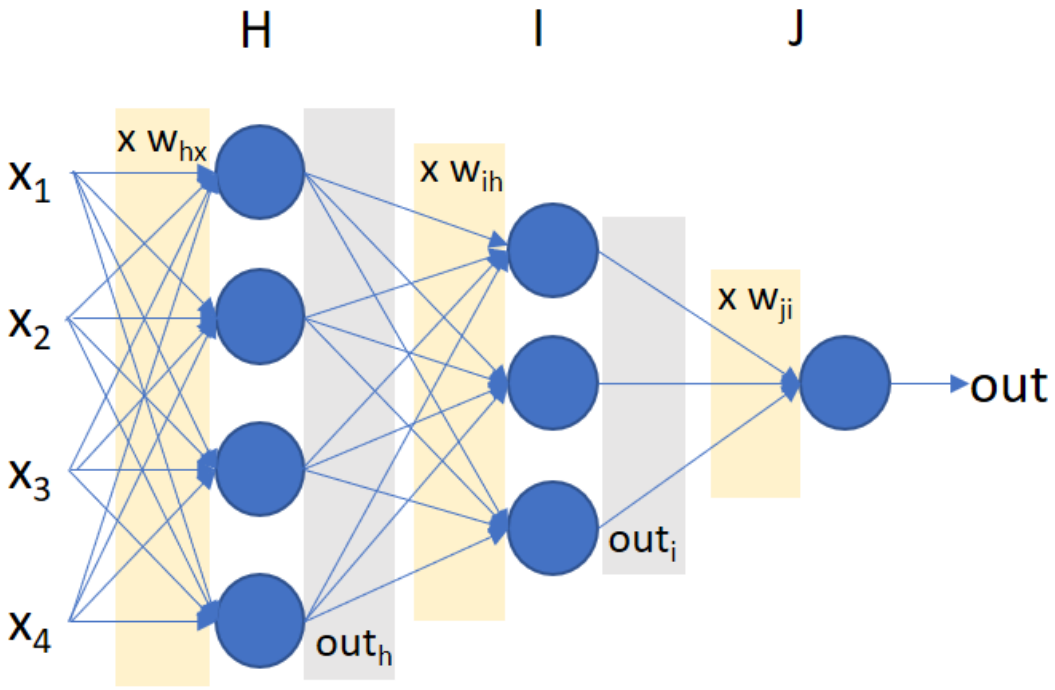

Figure 4.3 : Multi-layer neural network architecture.

decision tree GBDT proposed by (Hu and Min 2018). This additive regression model was introduced by (Friedman 2001) using a greedy algorithm called gradient boosting to optimize the model parameters namely the tree depth/levels, the number of models, the learning rate and the fraction of samples. It consists of multiple decision trees trained and tested in sequence as shown in Fig. 4.4. Each decision tree is producing an output score for a sample at the shaded node and the tree produces a total score for the sample by summing all the outputs(Ye et al. 2009). The GBDT training the model applying the negative gradients until the decision trees fit the model and produce correct output (Ke et al. 2017). To avoid overfitting, the GBDT comprises a parameter regulation improving the prediction accuracy (Hu and Min 2018).

Both multi-layer neural networks and the GBDT are implemented in the KNIME software for training and testing the classifiers with the input data from 3. To

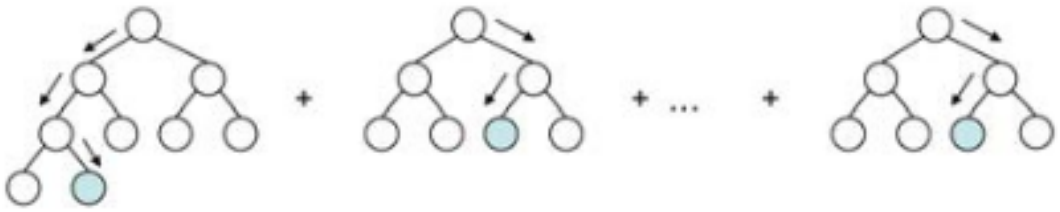

Figure 4.4 : GBDT ensemble of decision trees (Ye et al. 2009).

estimate the models' classification quality, the classifiers' sensitivity, specificity, F-measure, accuracy, cohen's kappa and ROC curve are detected and analyzed in KNIME.

1. Sensitivity

The rate of correctly classified positive (fatigue, label '1') samples to the total number of positive samples

$$Sensitivity = \frac{\#truepositive}{\#truepositive + \#falsenegative}\,. \tag{4.3}$$

2. Specificity

Likewise, the specificity is defined by the rate of correctly classified negative (normal, label '0') devided by the total number of negative samples

$$Specificity = \frac{\#truenegatives}{\#truenegatives + \#falsepositives}\,. \tag{4.4}$$

3. F-measure

This measurement focuses on the test's accuracy. It is composed of the value of precision p

$$p = \frac{\#truepositives}{\#truepositives + \#falsepositives} \tag{4.5}$$

and recall r which is equal to the Sensitivity. The F-measure or $F1$ score (Goutte and Gaussier 2005) is calculated from the harmonic mean

$$F1 = 2\frac{p\,r}{p+r}\,. \tag{4.6}$$

The ideal F-measure F1 = 1 is achieved with perfect precision $p = 1$ and recall $r = 1$.

4. Accuracy

The percentage of the correct predictions is called the classifier's accuracy

$$Accuracy = \frac{\#truepositives + \#truenegatives}{\#truepositives + \#truenegatives + \#falsepositives + \#falsenegatives}\,. \tag{4.7}$$

5. Cohen's kappa

This attribute describes the performance of the classifier compared to a randomly guessing classifier. If p_0 is the observed agreement and p_e is the expected agreement, then Cohen's kappa is defined as

$$K = \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e}\,. \tag{4.8}$$

6. ROC curve

A visual evaluation of the classifiers performance can be created by plotting the receiver operating characteristics (ROC) curve. On the x-axis, the true positive rate is displayed, and on the y-axis the false positive rate is plotted. Thus, the larger the area under the ROC curve, the higher is the accuracy of the classifier. This connection is also illustrated in Fig. 4.5.

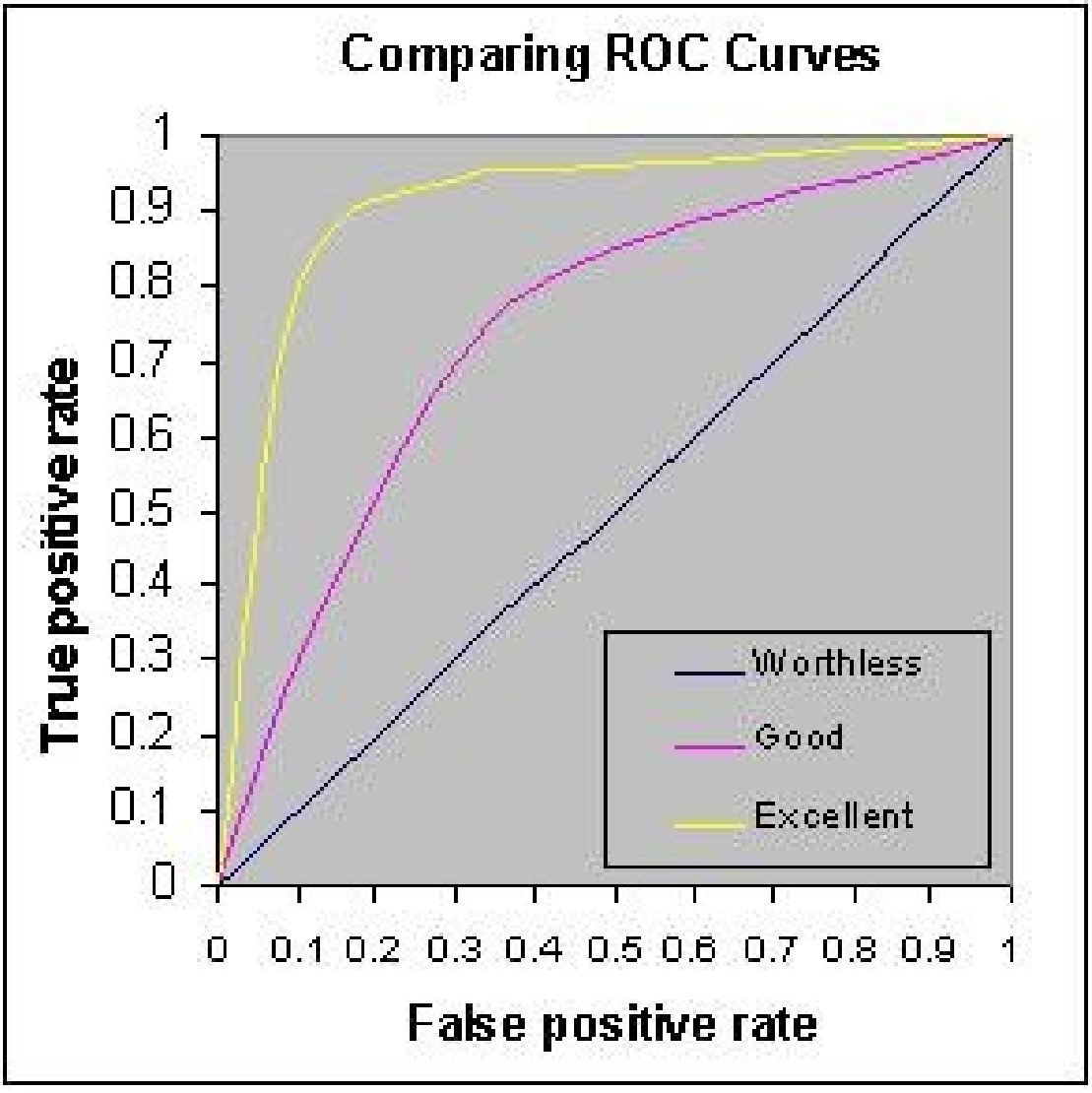

Figure 4.5 : ROC curves for a guessing, a good and an excellent classifier.

Chapter 5

Results

All test results of the implemented multi-layer neural network and the GBDT are summarized in the table in Fig. 5.1. Both models were trained and tested the same data samples from the feature set introduced in 3. From the results one can derive that the GBDT outperforms the multi-layer neural network and achieves excellent performance for the driver fatigue detection with an average accuracy of 93.62% on the validation test set. The ROC area under curve is very close to 1 which indicates excellent classification performance.

Some further preprocessing and preparation was applied to the features in KNIME before the data was fed into the classifiers. The full KNIME schematics for the logistic regression detector, the GBDT and the multi-layer neural network are plotted in Fig. 5.2. First, the feature dataset from 3 is loaded in a CSV reader. Next, the columns and labels are organized for the training. A column filter reduces

Feature Method	Classifier	EEG Channels	Sensitivity	Specificity	Accuracy Average (%)	Cohens Kappa	ROC result
Power Spectral Density, Standard Deviation, Mean, Sample Entropy	Multi-Layer Perception Neural Network	TP7	0.642	0.663	65.28	0.306	0.713
Statistical Features.	Logistic Regression ML	All Channels	0.532	0.558	75.02	0.502	0.820
Power Spectral Density, Standard Deviation, Mean, Sample Entropy	Multi-Layer Perception Neural Network	All Channels	0.829	0.826	81.3	0.625	0.893
Power Spectral Density, Standard Deviation, Mean, Sample Entropy, Fuzzy Entropy	Gradient Boosted Tree (GBT)	All Channels	0.918	0.924	93.62	0.872	0.982

Figure 5.1 : Table of test results for each implemented classifier.

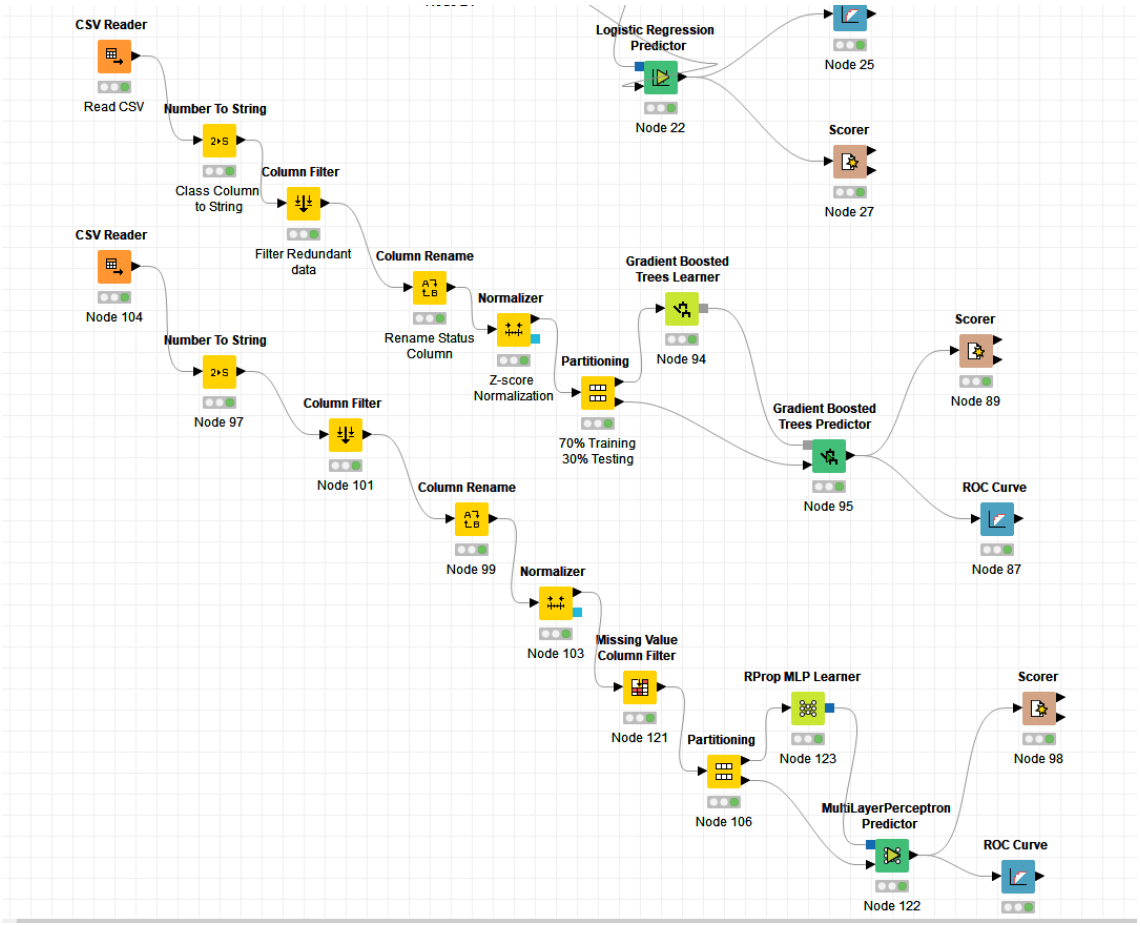

Figure 5.2 : Full KNIME schematic for the implemented logistic regression detector, GBDT and multi-layer neural network.

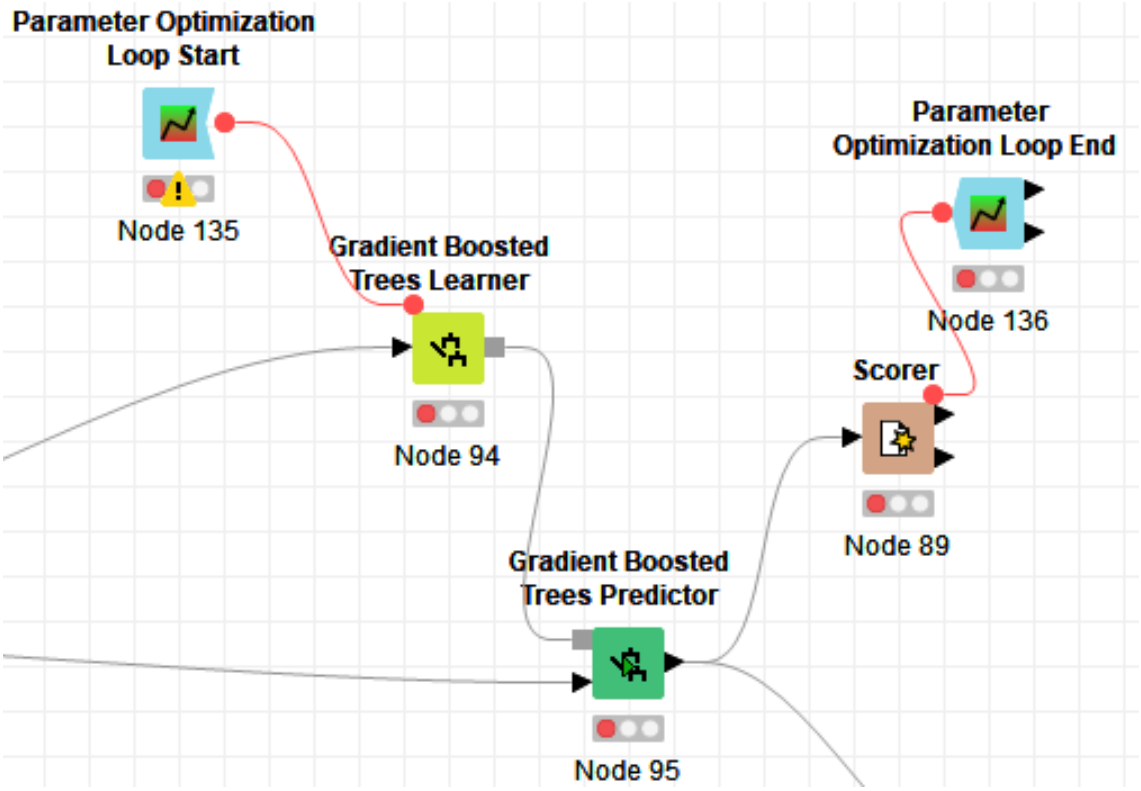

Figure 5.3 : KNIME schematic for the GBDT parameter optimization loop.

the dataset if there is redundant data for better training results. Next, the samples from the feature dataset are normalized applying gaussian z-score normalization. This method strives for setting standard deviation of the dataset to 1 and average to 0 which stabilizes the training process. After that, all samples are partitioned into training (70%) and testing (30%) and fed into the classifier. With GBDT, a parameter optimization loop shown in Fig. 5.3 is occupied between learner and scorer to compute the ideal settings for the GBDT parameters tree depth, number of trees and learning rate. After optimization, the optimal tree depth for the EEG feature set is detected to 4, the optimal number of models is 100 and the learning rate 0.1.

The GBDT's ROC curve is depicted in Fig. 5.4 to demonstrate the accuracy of the classifier. Another indication of a substential prediciton performance is the GBDT's confidence in the classification which lies around 93% and is presented in

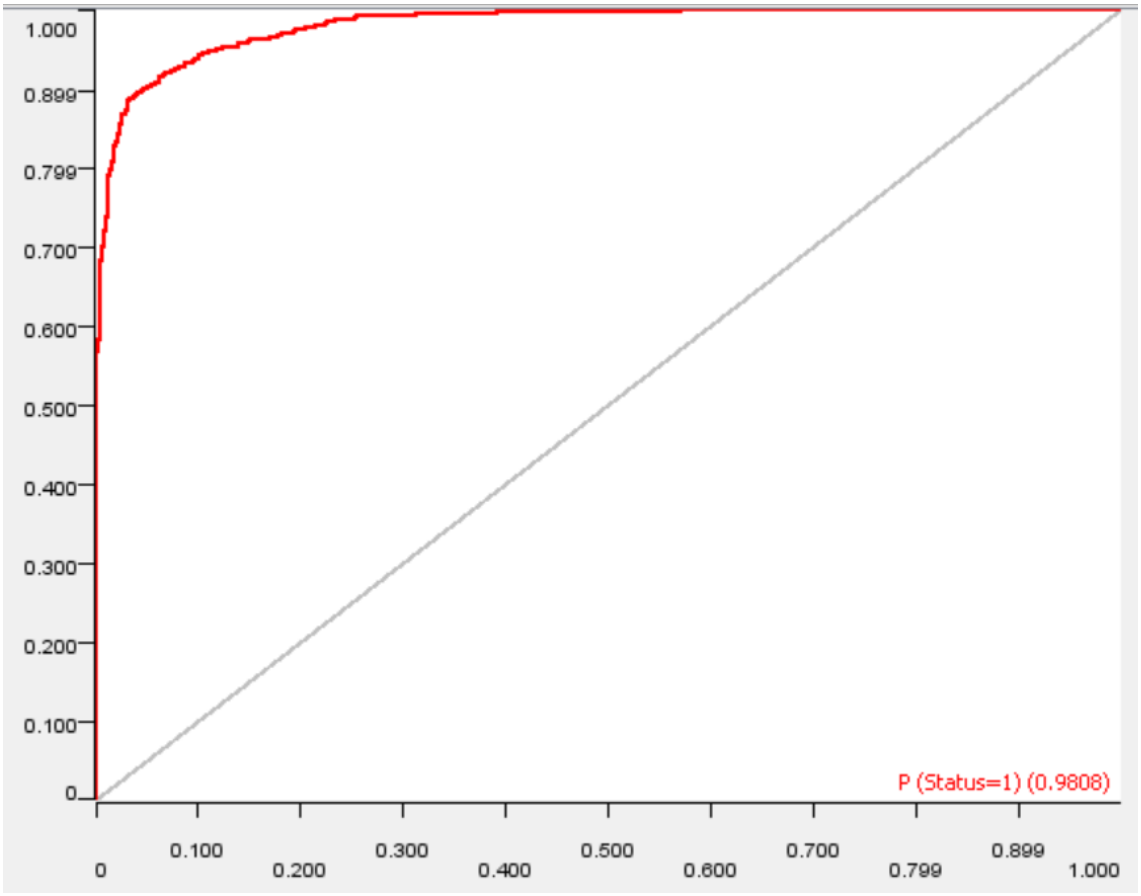

Figure 5.4 : ROC of the GBDT classifier.

the table in Fig. 5.5. For comparison, the multi-layer neural network's confidence on the same samples is shown in Fig. 5.6. It can be seen, that the neural network does not produce as clear decisions as the GBDT and therefore also produces more errors.

S Status	D P (Status=0)	D P (Status=1)	S Prediction (Status)	D Prediction (Status) (Confidence)
0	0.963	0.037	0	0.963
0	0.977	0.023	0	0.977
0	0.951	0.049	0	0.951
0	0.961	0.039	0	0.961
0	0.974	0.026	0	0.974
0	0.888	0.112	0	0.888
0	0.962	0.038	0	0.962
0	0.972	0.028	0	0.972
0	0.954	0.046	0	0.954
0	0.965	0.035	0	0.965
0	0.974	0.026	0	0.974
0	0.967	0.033	0	0.967
0	0.968	0.032	0	0.968
0	0.946	0.054	0	0.946
0	0.969	0.031	0	0.969
0	0.967	0.033	0	0.967
0	0.887	0.113	0	0.887
0	0.916	0.084	0	0.916
0	0.11	0.89	1	0.89
0	0.075	0.925	1	0.925
0	0.325	0.675	1	0.675

Figure 5.5 : Table of the GBDT classification results and confidence.

S Status	D P (Status=0)	D P (Status=1)	S Prediction (Status)
0			0
0			0
0			0
0			0
0			0
0			0
0			0
0			0
0			0
0			0
0			0
0			0
0			0
0			0
0			0
0			0
0			1
0			0
0			0
0			0
0			0
0			0
0			0
0			0
0			0
0			0
0			1
0			1

Figure 5.6 : Table of the multi-layer neural network classification results and confidence.

Chapter 6

Conclusion

It was shown, that the GBDT achieves excellent classification results when trained on the feature set containing PSD, signal mean, signal standard deviation, sample entropy and WPT coefficients selected with fuzzy entropy. Including all channels in the measurement achieved the highest performance. Using only a selection of channels like (Wang et al. 2019) reduces the accuracy but still gives sufficient accuracy. Since multiple measurements could be applied with a measurement duration of 1 second, the lower accuracy can be compensated and fatigue can be detected more accurately after multiple 1 second samples.

A collection of publications on driver fatigue presented in Fig. 6.1. Features, classification method and channels used can be compared with the average accuracy. The driver fatigue classification task is considered to be completed successfully with an overall excellent average accuracy of 93.62, a sensitivity of 0.918, specificity of 0.924 and a Cohen's kappa of 0.872.

To make the fatigue detection applicable inside a car, a reduction of complexity is desired. Scope of following research should be to implement an efficient driver fatigue detection using less EEG channels. Also, the number of features can be reduced by testing the features for significance and dropping the features carrying the lowest information about the driver's fatigue state. Using less features can also reduce the complexity of the classifier and make the detection faster.

Feature Method	Classifier	EEG Channels	Sensitivity	Specificity	Accuracy Average (%)	Cohens Kappa	ROC result
Power Spectral Density, Standard Deviation, Mean, Sample Entropy	Multi-Layer Perception Neural Network	TP7	0.642	0.663	65.28	0.306	0.713
Statistical Features.	Logistic Regression ML	All Channels	0.532	0.558	75.02	0.502	0.820
Power Spectral Density, Standard Deviation, Mean, Sample Entropy	Multi-Layer Perception Neural Network	All Channels	0.829	0.826	81.3	0.625	0.893
Power Spectral Density, Standard Deviation, Mean, Sample Entropy, Fuzzy Entropy	Gradient Boosted Tree (GBT)	All Channels	0.918	0.924	93.62	0.872	0.982

Figure 6.1 : Table of the previous publications on driver fatigue detection. Methods and results are compared.

Bibliography

Cao, Z. & Lin, C., 2018, 'Inherent fuzzy entropy for the improvement of eeg complexity evaluation', *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 2, pp. 1032–1035.

Chorzepa, M. G., Saeidpour, A., Christian, J. K. & Durham, S. A., 2016, 'Hurricane vulnerability of coastal bridges using multiple environmental parameters', *International Journal of Safety and Security Engineering*, vol. 6, no. 1, pp. 10–18.

Combes, J.-M., Grossmann, A. & Tchamitchian, P., 1990, *Wavelets: Time-Frequency Methods and Phase Space Proceedings of the International Conference, Marseille, France, December 14-18, 1987*, inverse problems and theoretical imaging, second edition edn., Springer, Berlin and Heidelberg.

Craig, A., Tran, Y., Wijesuriya, N. & Nguyen, H., 2012, 'Regional brain wave activity changes associated with fatigue', *Psychophysiology*, vol. 49, pp. 574–82.

Friedman, J. H., 2001, 'Greedy function approximation: A gradient boosting machine', *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, <`http://www.jstor.org/stable/2699986`>.

Goutte, C. & Gaussier, É., 2005, 'A probabilistic interpretation of precision, recall and f-score, with implication for evaluation', *ECIR*, .

Hu, J. & Min, J., 2018, 'Automated detection of driver fatigue based on eeg signals using gradient boosting decision tree model', *Cognitive Neurodynamics*, vol. 12, no. 4, pp. 431–440, <`https://doi.org/10.1007/s11571-018-9485-1`>.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. & Liu, T.-Y., 2017, 'Lightgbm: A highly efficient gradient boosting decision tree', *NIPS*, .

Khushaba, R. N., Kodagoda, S., Lal, S. & Dissanayake, G., 2011, 'Driver drowsiness classification using fuzzy wavelet-packet-based feature-extraction algorithm', *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 1, pp. 121–131.

Li, Z., Yang, Q., Chen, S., Zhou, W., Chen, L. & Song, L., 2019, 'A fuzzy recurrent neural network for driver fatigue detection based on steering-wheel angle sensor data', *International Journal of Distributed Sensor Networks*, vol. 15, no. 9, p. 1550147719872452, <https://doi.org/10.1177/1550147719872452>.

Min, J., Wang, P. & Hu, J., 2017, 'The original EEG data for driver fatigue detection', <https://figshare.com/articles/The_original_EEG_data_for_driver_fatigue_detection/5202739>.

Onton, J. & Makeig, S., 2006, 'Information-based modeling of event-related brain dynamics', Neuper, C. & Klimesch, W. (eds.) *Event-Related Dynamics of Brain Oscillations*, , vol. 159 of *Progress in Brain Research*Elsevier, pp. 99 – 120, <http://www.sciencedirect.com/science/article/pii/S0079612306590077>.

Subha, D. P., Joseph, P. K., Acharya U, R. & Lim, C. M., 2010, 'Eeg signal analysis: a survey', *Journal of medical systems*, vol. 34, no. 2, pp. 195–212.

TAC, 2018, 'Fatigue statistics: Road safety', viewed

Tran, Y., Thuraisingham, R. A., Wijesuriya, N., Nguyen, H. T. & Craig, A., 2007, 'Detecting neural changes during stress and fatigue effectively: a comparison of spectral analysis and sample entropy', *2007 3rd International IEEE/EMBS Conference on Neural Engineering*, pp. 350–353.

Wang, H., Wu, C., Li, T., He, Y., Chen, P. & Bezerianos, A., 2019, 'Driving fatigue

classification based on fusion entropy analysis combining eog and eeg', *IEEE Access*, vol. 7, pp. 61975–61986.

Ye, J., Chow, J.-H., Chen, J. & Zheng, Z., 2009, 'Stochastic gradient boosted distributed decision trees', *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, ACM, New York, NY, USA, pp. 2061–2064, <http://doi.acm.org/10.1145/1645953.1646301>.

Zahid Hasan, 2017, 'Perceptron from scratch in python', viewed