DL_feature_classifier

August 29, 2024

[1]:
```
import glob
import mne
import os
import re
from bs4 import BeautifulSoup
from keras import Sequential
from keras.src.callbacks import EarlyStopping, ReduceLROnPlateau
from keras.src.utils import to_categorical
from joblib import dump
from matplotlib import pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,␣
 ↪accuracy_score, classification_report
from sklearn.model_selection import train_test_split
import numpy as np
from scipy.stats import skew, kurtosis, entropy, mode
from scipy.signal import stft, welch
#from mne.time_frequency import psd_array_multitaper
#import antropy as ant  # For sample entropy and spectral entropy
#from pywt import wavedec
import antropy as ant
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization,␣
 ↪LeakyReLU
```

Ordner mit edf-Dateien suchen und vorbereiten

[2]:
```
def find_edf(directory):
    """
    Function to find the .edf files with ending [...] in a directory.
    """
    paths = []
    pattern = r".*\[\d+\].edf"
    for filename in os.listdir(directory):
        if re.match(pattern, filename):
            filepath = os.path.join(directory, filename)
            paths.append(filepath)

    paths.sort()
```

```
    return paths
```

Gewählte Kanäle aus EDF-Datei auslesen

[3]:
```
def files_preparation(path):
    """
    Read .edf File and choose channels
    """

    include = [
        'EEG C4-A1',
        'EEG C3-A2',
        'EOG ROC-A1',
        'EOG LOC-A2',
        'EMG Chin',
        'ECG I',
        'ECG II',
        'EEG A1-A2'
    ]

    raw = mne.io.read_raw_edf(path, preload=True, verbose='error',␣
 ↪include=include)

    raw.set_channel_types(
        mapping={
        'EEG C4-A1': 'eeg',
        'EEG C3-A2': 'eeg',
        'EEG A1-A2': 'eeg',
        'EOG ROC-A1': 'eog',
        'EOG LOC-A2': 'eog',
        'EMG Chin': 'emg',
        'ECG I': 'ecg',
        'ECG II': 'ecg'
        }
    )

    return raw
```

Labels aus der dazugehörigen .rml-Datei auslesen

[4]:
```
def rml_to_annotations(directory, raw_data):
    """
    Reads .rml file and converts labels to for MNE.Epochs useable onset,␣
 ↪describtion, duration.
    Where onset is the start of an Epoch, description is the label of the Epoch␣
 ↪and duration is the duration of the Epoch.
    """
```

```
    os.chdir(directory)
    rml = None
    for file in glob.glob("*.rml"):
        rml = file
        break
    with open(rml, 'r') as f:
        rml_data = f.read()
    user_staging = BeautifulSoup(rml_data, 'xml').find("UserStaging").
↪find("NeuroRKStaging")
    start_time = []
    sleep_stage = []
    for stage in user_staging.find_all('Stage'):
        start_time.append(int(stage['Start']))
        sleep_stage.append(stage['Type'])

    onset = np.array(start_time)
    description = np.array(sleep_stage)
    raw_duration = raw_data.times[-1] - raw_data.times[0]
    duration = np.diff(np.append(onset, raw_duration))
    return onset, description, duration
```

Erstellt 30 Sekunden Epochen aus allen EDF-Dateien und den Labels und Preprocesseses die rohen Dateien

[5]:
```
tmax = 30 - 1 / 200 # tmax describes length of an Epoch

# Sleep stages acording to AASM
EVENTS_AASM = {
    "REM": 1,
    "NREM 1": 2,
    "NREM 2": 3,
    "NREM 3": 4,
    "Wake": 5,
}

def data_preparation(directory):
    """
    Function converts the raw edf files to MNE.Epochs and returns them.
    """

    print("Data from Directory:  ", directory)
    paths = find_edf(directory)
    data_list = []
    for path in paths:
        raw = files_preparation(path)
        data_list.append(raw)
```

```
    raw_data = mne.concatenate_raws(data_list)


    raw_data = raw_data.filter(
        picks='all',
        l_freq=0.5,
        h_freq=49.5,
        method='iir',
        iir_params=dict(order=10, ftype='butter'),
        verbose='error'
    )


    onset, description, duration = rml_to_annotations(directory, raw_data)
    annotations = mne.Annotations(onset=onset, description=description,␣
↪duration=duration)
    annotations.crop(
        annotations[1]['onset'] - 30 * 20, # 30 * 60 = 1200 Ersten 10 Minuten␣
↪entfernen
        annotations[-2]['onset'] + 30 * 20 # Letzten 10 Minuten entfernen
    )
    raw_data.set_annotations(annotations)

    events, _ = mne.events_from_annotations(
        raw_data,
        chunk_duration=30,
        verbose='error'
    )

    epochs = mne.Epochs(
        raw=raw_data,
        events=events,
        event_id=EVENTS_AASM,
        tmin=0.0,
        baseline=None,
        tmax=tmax,
        verbose='error',
        on_missing='warn'
    )

    labels = epochs.events[:, 2]

    return epochs, labels
```

Feature Extraction 1. Time Domain Features

[6]:
```python
def mean(epochs):
    # Calculates Mean
    data = epochs.get_data(verbose='error')
    mean_features = np.mean(data, axis=2)
    return mean_features

def median(epochs):
    # Calculates Median
    data = epochs.get_data(verbose='error')
    median_features = np.median(data, axis=2)
    return median_features

def mode_feat(epochs):
    # Calculates Mode
    data = epochs.get_data(verbose='error')
    mode_features, _ = mode(data, axis=2)
    return mode_features

def minimum(epochs):
    # Calculates Minimum
    data = epochs.get_data(verbose='error')
    min_values = np.min(data, axis=2)
    return min_values

def maximum(epochs):
    # Calculates Minimum
    data = epochs.get_data(verbose='error')
    return np.max(data, axis=2)

def standard_derivation(epochs):
    # Calculates standard deviation
    data = epochs.get_data(verbose='error')
    std_features = np.std(data, axis=2)
    return std_features

def variance_skewness(epochs):
    # Calculates Variance Skewness
    data = epochs.get_data(verbose='error')
    variance_features = np.var(data, axis=2)
    return variance_features

def kurtosis_feat(epochs):
    # Calculates Kurtosis
    data = epochs.get_data(verbose='error')
    kurtosis_data = kurtosis(data, axis=2, fisher=False)
    return kurtosis_data
```

```python
def percentile(epochs, percent = 50):
    # Calculates Percentile
    data = epochs.get_data(verbose='error')
    percentile_features = np.percentile(data, percent, axis=2)
    return percentile_features

def energy_sis(epochs):
    """
    Calculates the energy of the signals
    """

    data = epochs.get_data(verbose='error')
    n_epochs, n_channels, n_times = data.shape

    energy_array = np.zeros((n_epochs, n_channels))

    for epoch_idx in range(n_epochs):
        for channel_idx in range(n_channels):
            signal = data[epoch_idx, channel_idx, :]

            energy = np.sum(signal ** 2)

            energy_array[epoch_idx, channel_idx] = energy

    return energy_array


def time_domain_features(epochs):
    """
    Where all time domain functions are called
    """
    time_features_methods = [
        # Hier werden die Time Domain Feature berechnet
        mean(epochs),
        median(epochs),
        mode_feat(epochs),
        minimum(epochs),
        maximum(epochs),
        standard_derivation(epochs),
        variance_skewness(epochs),
        kurtosis_feat(epochs),
        percentile(epochs),
        energy_sis(epochs)

    ]
    return np.concatenate(time_features_methods, axis=1)
```

2. Frequency Domain Features

[7]:
```
def eeg_power_band(epochs):
    """
    Calculates power bands from EEG channels
    """

    FREQ_BANDS = {
        "delta": [0.5, 4],
        "theta": [4, 8],
        "alpha": [8, 13],
        "sigma": [12, 16],
        "beta": [13, 30],
    }

    spectrum = epochs.compute_psd(picks="eeg", method='welch', fmin=0.5,␣
 ↪fmax=30.0, verbose='error')
    psds, freqs = spectrum.get_data(return_freqs=True)
    psds /= np.sum(psds, axis=-1, keepdims=True)

    X = []
    for fmin, fmax in FREQ_BANDS.values():
        psds_band = psds[:, :, (freqs >= fmin) & (freqs < fmax)].mean(axis=-1)
        X.append(psds_band.reshape(len(psds), -1))

    re = np.concatenate(X, axis=1)
    return re


def frequency_domain_features(epochs):
    """
    Where all frequency domain functions are called
    """
    frequency_features_methods = [
        eeg_power_band(epochs),
    ]
    return np.concatenate(frequency_features_methods, axis=1)
```

3. Non-Linear Features

[8]:
```
def zero_cross_rate(epochs):
    # Calculates Zero Cross Rate
    data = epochs.get_data(verbose='error')
    return np.mean(np.diff(np.sign(data), axis=2) != 0, axis=2)

def spectral_entropy(epochs, method='fft', **kwargs):
    """
    Calculates the spectral entropy for each epoch and channel
    """
```

```
    data = epochs.get_data(verbose='error')
    sfreq = epochs.info['sfreq']
    n_epochs, n_channels, n_times = data.shape

    spec_entropy_array = np.zeros((n_epochs, n_channels))

    for epoch_idx in range(n_epochs):
        for channel_idx in range(n_channels):
            single_epoch_data = data[epoch_idx, channel_idx, :]
            se = ant.spectral_entropy(single_epoch_data, method=method,␣
 ↪sf=sfreq, **kwargs)
            spec_entropy_array[epoch_idx, channel_idx] = se

    return spec_entropy_array

def non_linear_features(epochs):
    """
    Where all non-linear functions are called
    """
    nlinear_features_methods = [
        zero_cross_rate(epochs),
        spectral_entropy(epochs)
    ]
    return np.concatenate(nlinear_features_methods, axis=1)
```

Feature Function

[9]:
```
def feature_extraction(epochs):
    """
    Calls the functions that calculate the features and returns np.Array␣
 ↪containing the features
    """
    features_methods = [
        time_domain_features(epochs),
        frequency_domain_features(epochs),
        non_linear_features(epochs)
    ]

    features_methods = np.concatenate(features_methods, axis=1)
    return features_methods
```

Sucht nach allen Unterordner, liest diese aus und gibt fertige Epochen aus allen Dateien zurück

[10]:
```
def list_subdirectories(directory):
    """
    Lists all subdirectories in the given directory.
```

```
    """
    return [os.path.join(directory, sub_dir) for sub_dir in os.
 ↪listdir(directory) if os.path.isdir(os.path.join(directory, sub_dir))]

def process_all_folders(main_directory):
    """
    Iterates over all subdirectories in the main directory and prepares data.
    """
    all_epochs = []
    all_labels = []

    subdirectories = list_subdirectories(main_directory)

    for sub_dir in subdirectories:
        epochs, labels = data_preparation(sub_dir)
        epochs = feature_extraction(epochs)
        all_epochs.append(epochs)
        all_labels.append(labels)

    # Combine all data if needed
    combined_epochs = np.concatenate(all_epochs, axis=0) if all_epochs else None
    combined_labels = np.concatenate(all_labels, axis=0) if all_labels else None

    return combined_epochs, combined_labels


# Directory
main_directory = '/Volumes/Jonas_SSD/test'

# Saves Array with data and labels in X and y
X, y = process_all_folders(main_directory)
```

```
Data from Directory:   /Volumes/Jonas_SSD/test/00000021-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000702-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000042-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000775-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000035-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000398-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000062-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000706-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000043-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000055-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000708-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000060-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000077-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000709-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000070-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000087-A5BS00755
```

```
Data from Directory:   /Volumes/Jonas_SSD/test/00000710-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000079-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000080-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000719-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000086-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000089-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000726-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000091-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000094-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000727-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000106-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000107-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000728-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000108-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000110-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000731-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000111-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000113-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000736-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000114-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000115-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000743-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000116-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000117-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000744-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000123-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000124-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000751-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000126-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000127-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000752-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000133-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000136-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000753-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000137-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000139-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000755-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000145-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000147-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000759-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000152-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000153-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000767-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000160-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000161-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000769-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000169-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000180-A5BS00755
```

```
Data from Directory:   /Volumes/Jonas_SSD/test/00000770-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000181-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000193-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000772-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000199-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000204-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000774-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000206-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000207-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000211-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000213-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000216-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000223-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000231-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000233-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000242-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000243-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000245-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000246-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000248-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000263-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000273-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000283-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000284-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000285-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000287-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000293-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000297-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000299-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000300-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000319-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000321-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000322-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000330-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000331-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000342-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000351-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000352-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000357-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000359-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000368-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000369-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000378-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000380-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000384-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000385-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000386-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000395-A5BS00755
```

```
Data from Directory:   /Volumes/Jonas_SSD/test/00000396-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000403-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000416-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000423-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000424-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000425-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000427-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000431-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000432-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000441-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000448-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000458-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000466-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000480-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000493-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000494-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000496-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000500-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000501-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000509-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000512-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000515-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000516-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000517-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000519-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000523-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000527-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000529-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000531-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000534-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000536-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000538-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000542-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000543-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000544-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000546-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000548-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000550-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000551-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000552-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000553-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000561-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000562-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000574-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000576-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000578-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000586-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000597-A5BS00755
```

```
Data from Directory:   /Volumes/Jonas_SSD/test/00000605-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000614-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000645-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000649-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000657-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000658-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000666-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000674-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000676-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000685-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000686-A5BS00755
Data from Directory:   /Volumes/Jonas_SSD/test/00000687-A5BS00755
```

[11]:
```
X.shape, y.shape
```

[11]:
```
((129754, 111), (129754,))
```

Modell vorbereiten, trainieren und testen

Alle EEG, ECG & EOG:

- 63,86% -> 2 Hidden Layer mit ReLu und 64 Notes - 30 Epochen - Klassen Acc ausgeglichen (58+)- kein Scaler - kein ReliefF - kein Dropout
- 74,84% -> 2 Hidden Layer mit ReLu und 64 Notes - 30 Epochen - Klassen Acc ausgeglichen (70+) - mit Scaler - kein ReliefF - kein Dropout
- 73,93% -> 2 Hidden Layer mit ReLu und 64 Notes - 30 Epochen - Klassen Acc ausgeglichen (68+) - mit Scaler - kein ReliefF - mit Dropout (30%)
- 75,71% -> 2 Hidden Layer mit ReLu und 64 Notes - 50 Epochen - Klassen Acc ausgeglichen (69+) - mit Scaler - kein ReliefF - mit Dropout (20%)
- 77,28% -> 3 Hidden Layer mit Relu und 128(2x), 64 Notes - 50 Epochen - Klassen Acc ausgeglichen (69%) - mit Scaler - kein ReliefF - mit Dropout (2x20%)
- 75,60% -> 3 Hidden Layer mit Relu und 128, 64, 32 Notes - 50 Epochen - Klassen Acc ausgeglichen (68%) - mit Scaler - kein ReliefF - mit Dropout (2x20%)
- 76,79% -> 4 Hidden Layer mit ReLu und 128(2x), 64 ,32 Notes - 50 Epochen - Klassen Acc ausgeglichen (69+) - mit Scaler - kein ReliefF - mit Dropout (3x20%)
- 77,92% -> 3 Hidden Layer mit ReLu und 256, 128, 64 Notes - 50 Epochen - Klassen Acc ausgeglichen (71+) - mit Scaler - kein ReliefF - mit Dropout (2x20%)
- 78,06% -> 3 Hidden Layer mit ReLu und 256, 128, 64 Notes - 50 Epochen - Klassen Acc ausgeglichen (69+) - mit Scaler - kein ReliefF - mit Dropout (2x20%) - mit Batch Normalization
- 78,27% -> 4 Hidden Layer mit ReLu und 256(2x), 128, 64 Notes - 80 Epochen - Klassen Acc ausgeglichen (71+) - mit Scaler - kein ReliefF - mit Dropout (30%, 2x20%) - kein Batch Normalization
- 78,16% -> 4 Hidden Layer mit ReLu und 512, 256, 128, 64 Notes - 80 Epochen - Klassen Acc ausgeglichen (71+) - mit Scaler - kein ReliefF - mit Dropout (30%, 2x20%) - kein Batch Normalization
- 78,58% -> 4 Hidden Layer mit ReLu und 256(2x), 128, 64 Notes - 80 Epochen - Klassen Acc ausgeglichen (72+) - mit Scaler - kein ReliefF - mit Dropout (30%, 2x20%) - kein Batch Normalization - mit Reduce LR on Plateau
- 78,76% -> 4 Hidden Layer mit ReLu und 256(2x), 128, 64 Notes - 80 Epochen - Klassen

Acc ausgeglichen (72+) - mit Scaler - kein ReliefF - mit Dropout (3x30%) - kein Batch Normalization - mit Reduce LR on Plateau

- 78,62% -> 4 Hidden Layer mit ReLu und 256(2x), 128, 64 Notes - 80 Epochen - Klassen Acc ausgeglichen (72+) - mit Scaler - kein ReliefF - mit Dropout (3x30%) - kein Batch Normalization
- 78,93% -> 4 Hidden Layer mit ReLu und 288, 416, 256, 64 Notes - 80 Epochen - Klassen Acc ausgeglichen (73+) -mit Scaler - kein ReliefF - mit Dropout (3x30%) - mit Batch Normalization nach dem ersten Layer
- 79,22% -> 3 Hidden Layer mit ReLu und 320, 512, 96 Notes - 80 Epochen - Klassen Acc ausgeglichen (73+) -mit Scaler - kein ReliefF - mit Dropout (3x30%) - mit Batch Normalization nach dem ersten Layer

Normalization

[12]:
```
print(f"Before Scaling: {X[222, :10]}")
X = StandardScaler().fit_transform(X)
print(f"After Scaling:  {X[222, :10]}")
```

```
Before Scaling: [ 2.64504821e-08 -1.64480549e-08  6.15684526e-08 -4.74998777e-09
 -1.11762382e-09  2.17014702e-09  1.32438149e-07 -2.54186287e-08
 -1.73766874e-07  5.28585550e-08]
After Scaling:  [ 0.2242799  -0.14475999  0.50272108 -0.03499775 -0.01375852
0.00717116
  0.19499255 -0.06149885 -0.68829066  0.04905104]
```

One-Hot Encoding

[13]:
```
print(f"Alte Labelnummern: {np.unique(y)}")
EVENTS_AASM_NEU = {
    "REM": 0,
    "NREM 1": 1,
    "NREM 2": 2,
    "NREM 3": 3,
    "Wake": 4,
}

y = y - 1
print(f"Neue Labelnummern: {np.unique(y)}")
y = to_categorical(y)
```

```
Alte Labelnummern: [1 2 3 4 5]
Neue Labelnummern: [0 1 2 3 4]
```

Splitting data in train and test set

[29]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

ANN Model

[30]:
```
#model = Sequential()
#model.add(Dense(512, activation='relu'))
#model.add(Dropout(0.5))
#model.add(BatchNormalization())
#model.add(Dense(256, activation='relu'))
#model.add(Dropout(0.4))
#model.add(Dense(128, activation='relu'))
#model.add(Dropout(0.3))
#model.add(Dense(64, activation='relu'))
#model.add(Dense(5, activation='softmax'))


#model = Sequential()
#model.add(Dense(384, activation='relu'))
#model.add(Dropout(0.40))
#model.add(BatchNormalization())
#model.add(Dense(512, activation='relu'))
#model.add(Dropout(0.35))
#model.add(Dense(256, activation='relu'))
##model.add(Dropout(0.30))
#model.add(Dense(128, activation='relu'))
#model.add(Dropout(0.30))
#model.add(Dense(64, activation='relu'))
#model.add(Dense(5, activation='softmax'))

model = Sequential()
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.35))
model.add(BatchNormalization())
model.add(Dense(448, activation='relu'))
model.add(Dropout(0.35))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(5, activation='softmax'))



model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])


early_stopping = EarlyStopping(monitor='val_loss', patience=8,␣
 ↪restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3,␣
 ↪min_lr=0.00001, verbose=1)
```

```
history = model.fit(X_train, y_train, epochs=80, validation_split=0.15,␣
 ↪callbacks=[reduce_lr], verbose=2)
```

```
Epoch 1/80
2758/2758 - 10s - loss: 0.9796 - accuracy: 0.6021 - val_loss: 0.8168 -
val_accuracy: 0.6738 - lr: 0.0010 - 10s/epoch - 4ms/step
Epoch 2/80
2758/2758 - 7s - loss: 0.8727 - accuracy: 0.6483 - val_loss: 0.7475 -
val_accuracy: 0.6926 - lr: 0.0010 - 7s/epoch - 3ms/step
Epoch 3/80
2758/2758 - 7s - loss: 0.8329 - accuracy: 0.6630 - val_loss: 0.7266 -
val_accuracy: 0.7019 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 4/80
2758/2758 - 7s - loss: 0.8034 - accuracy: 0.6735 - val_loss: 0.7061 -
val_accuracy: 0.7141 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 5/80
2758/2758 - 7s - loss: 0.7808 - accuracy: 0.6814 - val_loss: 0.6738 -
val_accuracy: 0.7251 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 6/80
2758/2758 - 7s - loss: 0.7661 - accuracy: 0.6877 - val_loss: 0.6669 -
val_accuracy: 0.7260 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 7/80
2758/2758 - 7s - loss: 0.7502 - accuracy: 0.6933 - val_loss: 0.6511 -
val_accuracy: 0.7224 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 8/80
2758/2758 - 6s - loss: 0.7409 - accuracy: 0.6983 - val_loss: 0.6540 -
val_accuracy: 0.7298 - lr: 0.0010 - 6s/epoch - 2ms/step
Epoch 9/80
2758/2758 - 6s - loss: 0.7288 - accuracy: 0.7004 - val_loss: 0.6488 -
val_accuracy: 0.7239 - lr: 0.0010 - 6s/epoch - 2ms/step
Epoch 10/80
2758/2758 - 7s - loss: 0.7221 - accuracy: 0.7030 - val_loss: 0.6513 -
val_accuracy: 0.7310 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 11/80
2758/2758 - 7s - loss: 0.7182 - accuracy: 0.7054 - val_loss: 0.6352 -
val_accuracy: 0.7395 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 12/80
2758/2758 - 7s - loss: 0.7132 - accuracy: 0.7064 - val_loss: 0.6314 -
val_accuracy: 0.7436 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 13/80
2758/2758 - 7s - loss: 0.7046 - accuracy: 0.7105 - val_loss: 0.6438 -
val_accuracy: 0.7421 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 14/80
2758/2758 - 7s - loss: 0.7001 - accuracy: 0.7121 - val_loss: 0.6113 -
val_accuracy: 0.7447 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 15/80
```

```
2758/2758 - 7s - loss: 0.6951 - accuracy: 0.7141 - val_loss: 0.6195 -
val_accuracy: 0.7411 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 16/80
2758/2758 - 7s - loss: 0.6926 - accuracy: 0.7154 - val_loss: 0.6114 -
val_accuracy: 0.7425 - lr: 0.0010 - 7s/epoch - 3ms/step
Epoch 17/80
2758/2758 - 7s - loss: 0.6885 - accuracy: 0.7160 - val_loss: 0.6059 -
val_accuracy: 0.7473 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 18/80
2758/2758 - 7s - loss: 0.6876 - accuracy: 0.7165 - val_loss: 0.6033 -
val_accuracy: 0.7447 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 19/80
2758/2758 - 6s - loss: 0.6782 - accuracy: 0.7216 - val_loss: 0.6007 -
val_accuracy: 0.7506 - lr: 0.0010 - 6s/epoch - 2ms/step
Epoch 20/80
2758/2758 - 7s - loss: 0.6778 - accuracy: 0.7221 - val_loss: 0.5939 -
val_accuracy: 0.7525 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 21/80
2758/2758 - 7s - loss: 0.6752 - accuracy: 0.7227 - val_loss: 0.5986 -
val_accuracy: 0.7542 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 22/80
2758/2758 - 7s - loss: 0.6673 - accuracy: 0.7269 - val_loss: 0.6045 -
val_accuracy: 0.7523 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 23/80
2758/2758 - 7s - loss: 0.6685 - accuracy: 0.7250 - val_loss: 0.5886 -
val_accuracy: 0.7524 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 24/80
2758/2758 - 7s - loss: 0.6649 - accuracy: 0.7261 - val_loss: 0.5932 -
val_accuracy: 0.7563 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 25/80
2758/2758 - 7s - loss: 0.6627 - accuracy: 0.7273 - val_loss: 0.6017 -
val_accuracy: 0.7475 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 26/80
2758/2758 - 7s - loss: 0.6598 - accuracy: 0.7264 - val_loss: 0.5870 -
val_accuracy: 0.7603 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 27/80
2758/2758 - 7s - loss: 0.6572 - accuracy: 0.7305 - val_loss: 0.5777 -
val_accuracy: 0.7618 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 28/80
2758/2758 - 7s - loss: 0.6567 - accuracy: 0.7307 - val_loss: 0.5816 -
val_accuracy: 0.7561 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 29/80
2758/2758 - 7s - loss: 0.6531 - accuracy: 0.7305 - val_loss: 0.5792 -
val_accuracy: 0.7623 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 30/80
2758/2758 - 7s - loss: 0.6533 - accuracy: 0.7318 - val_loss: 0.5757 -
val_accuracy: 0.7585 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 31/80
```

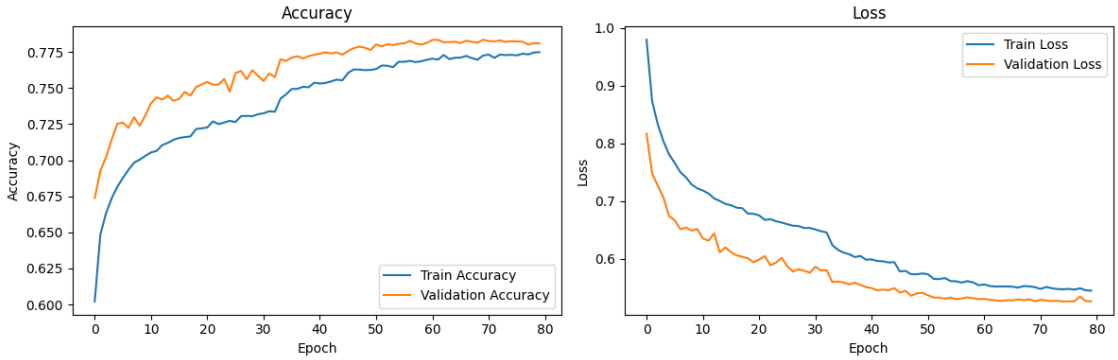
```
2758/2758 - 7s - loss: 0.6506 - accuracy: 0.7325 - val_loss: 0.5861 -
val_accuracy: 0.7549 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 32/80
2758/2758 - 7s - loss: 0.6476 - accuracy: 0.7339 - val_loss: 0.5797 -
val_accuracy: 0.7601 - lr: 0.0010 - 7s/epoch - 2ms/step
Epoch 33/80

Epoch 33: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
2758/2758 - 7s - loss: 0.6455 - accuracy: 0.7335 - val_loss: 0.5800 -
val_accuracy: 0.7576 - lr: 0.0010 - 7s/epoch - 3ms/step
Epoch 34/80
2758/2758 - 7s - loss: 0.6235 - accuracy: 0.7426 - val_loss: 0.5596 -
val_accuracy: 0.7699 - lr: 5.0000e-04 - 7s/epoch - 2ms/step
Epoch 35/80
2758/2758 - 7s - loss: 0.6155 - accuracy: 0.7456 - val_loss: 0.5604 -
val_accuracy: 0.7688 - lr: 5.0000e-04 - 7s/epoch - 2ms/step
Epoch 36/80
2758/2758 - 7s - loss: 0.6106 - accuracy: 0.7494 - val_loss: 0.5588 -
val_accuracy: 0.7711 - lr: 5.0000e-04 - 7s/epoch - 2ms/step
Epoch 37/80
2758/2758 - 7s - loss: 0.6077 - accuracy: 0.7494 - val_loss: 0.5554 -
val_accuracy: 0.7720 - lr: 5.0000e-04 - 7s/epoch - 2ms/step
Epoch 38/80
2758/2758 - 7s - loss: 0.6029 - accuracy: 0.7509 - val_loss: 0.5585 -
val_accuracy: 0.7705 - lr: 5.0000e-04 - 7s/epoch - 2ms/step
Epoch 39/80
2758/2758 - 7s - loss: 0.6047 - accuracy: 0.7506 - val_loss: 0.5544 -
val_accuracy: 0.7721 - lr: 5.0000e-04 - 7s/epoch - 2ms/step
Epoch 40/80
2758/2758 - 7s - loss: 0.5982 - accuracy: 0.7537 - val_loss: 0.5508 -
val_accuracy: 0.7730 - lr: 5.0000e-04 - 7s/epoch - 2ms/step
Epoch 41/80
2758/2758 - 7s - loss: 0.5989 - accuracy: 0.7531 - val_loss: 0.5490 -
val_accuracy: 0.7739 - lr: 5.0000e-04 - 7s/epoch - 2ms/step
Epoch 42/80
2758/2758 - 7s - loss: 0.5961 - accuracy: 0.7535 - val_loss: 0.5451 -
val_accuracy: 0.7747 - lr: 5.0000e-04 - 7s/epoch - 2ms/step
Epoch 43/80
2758/2758 - 7s - loss: 0.5955 - accuracy: 0.7546 - val_loss: 0.5463 -
val_accuracy: 0.7740 - lr: 5.0000e-04 - 7s/epoch - 2ms/step
Epoch 44/80
2758/2758 - 7s - loss: 0.5934 - accuracy: 0.7558 - val_loss: 0.5454 -
val_accuracy: 0.7746 - lr: 5.0000e-04 - 7s/epoch - 2ms/step
Epoch 45/80

Epoch 45: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
2758/2758 - 7s - loss: 0.5940 - accuracy: 0.7553 - val_loss: 0.5490 -
val_accuracy: 0.7731 - lr: 5.0000e-04 - 7s/epoch - 2ms/step
```

```
Epoch 46/80
2758/2758 - 7s - loss: 0.5780 - accuracy: 0.7604 - val_loss: 0.5414 -
val_accuracy: 0.7757 - lr: 2.5000e-04 - 7s/epoch - 2ms/step
Epoch 47/80
2758/2758 - 7s - loss: 0.5789 - accuracy: 0.7629 - val_loss: 0.5444 -
val_accuracy: 0.7776 - lr: 2.5000e-04 - 7s/epoch - 2ms/step
Epoch 48/80
2758/2758 - 7s - loss: 0.5732 - accuracy: 0.7628 - val_loss: 0.5356 -
val_accuracy: 0.7787 - lr: 2.5000e-04 - 7s/epoch - 2ms/step
Epoch 49/80
2758/2758 - 6s - loss: 0.5731 - accuracy: 0.7624 - val_loss: 0.5401 -
val_accuracy: 0.7779 - lr: 2.5000e-04 - 6s/epoch - 2ms/step
Epoch 50/80
2758/2758 - 7s - loss: 0.5744 - accuracy: 0.7625 - val_loss: 0.5409 -
val_accuracy: 0.7764 - lr: 2.5000e-04 - 7s/epoch - 2ms/step
Epoch 51/80

Epoch 51: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
2758/2758 - 7s - loss: 0.5731 - accuracy: 0.7632 - val_loss: 0.5366 -
val_accuracy: 0.7802 - lr: 2.5000e-04 - 7s/epoch - 2ms/step
Epoch 52/80
2758/2758 - 7s - loss: 0.5651 - accuracy: 0.7656 - val_loss: 0.5328 -
val_accuracy: 0.7789 - lr: 1.2500e-04 - 7s/epoch - 2ms/step
Epoch 53/80
2758/2758 - 9s - loss: 0.5647 - accuracy: 0.7654 - val_loss: 0.5325 -
val_accuracy: 0.7804 - lr: 1.2500e-04 - 9s/epoch - 3ms/step
Epoch 54/80
2758/2758 - 7s - loss: 0.5665 - accuracy: 0.7645 - val_loss: 0.5305 -
val_accuracy: 0.7798 - lr: 1.2500e-04 - 7s/epoch - 2ms/step
Epoch 55/80
2758/2758 - 6s - loss: 0.5612 - accuracy: 0.7682 - val_loss: 0.5326 -
val_accuracy: 0.7807 - lr: 1.2500e-04 - 6s/epoch - 2ms/step
Epoch 56/80
2758/2758 - 6s - loss: 0.5608 - accuracy: 0.7682 - val_loss: 0.5299 -
val_accuracy: 0.7809 - lr: 1.2500e-04 - 6s/epoch - 2ms/step
Epoch 57/80
2758/2758 - 6s - loss: 0.5586 - accuracy: 0.7688 - val_loss: 0.5311 -
val_accuracy: 0.7827 - lr: 1.2500e-04 - 6s/epoch - 2ms/step
Epoch 58/80
2758/2758 - 6s - loss: 0.5612 - accuracy: 0.7680 - val_loss: 0.5330 -
val_accuracy: 0.7809 - lr: 1.2500e-04 - 6s/epoch - 2ms/step
Epoch 59/80

Epoch 59: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
2758/2758 - 7s - loss: 0.5590 - accuracy: 0.7686 - val_loss: 0.5315 -
val_accuracy: 0.7801 - lr: 1.2500e-04 - 7s/epoch - 2ms/step
Epoch 60/80
2758/2758 - 6s - loss: 0.5540 - accuracy: 0.7696 - val_loss: 0.5300 -
```

```
val_accuracy: 0.7813 - lr: 6.2500e-05 - 6s/epoch - 2ms/step
Epoch 61/80
2758/2758 - 7s - loss: 0.5555 - accuracy: 0.7703 - val_loss: 0.5303 -
val_accuracy: 0.7834 - lr: 6.2500e-05 - 7s/epoch - 2ms/step
Epoch 62/80
2758/2758 - 7s - loss: 0.5526 - accuracy: 0.7698 - val_loss: 0.5287 -
val_accuracy: 0.7834 - lr: 6.2500e-05 - 7s/epoch - 3ms/step
Epoch 63/80
2758/2758 - 7s - loss: 0.5517 - accuracy: 0.7729 - val_loss: 0.5278 -
val_accuracy: 0.7817 - lr: 6.2500e-05 - 7s/epoch - 2ms/step
Epoch 64/80
2758/2758 - 7s - loss: 0.5519 - accuracy: 0.7700 - val_loss: 0.5269 -
val_accuracy: 0.7818 - lr: 6.2500e-05 - 7s/epoch - 2ms/step
Epoch 65/80
2758/2758 - 7s - loss: 0.5519 - accuracy: 0.7710 - val_loss: 0.5281 -
val_accuracy: 0.7821 - lr: 6.2500e-05 - 7s/epoch - 2ms/step
Epoch 66/80
2758/2758 - 6s - loss: 0.5516 - accuracy: 0.7711 - val_loss: 0.5280 -
val_accuracy: 0.7812 - lr: 6.2500e-05 - 6s/epoch - 2ms/step
Epoch 67/80

Epoch 67: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
2758/2758 - 7s - loss: 0.5498 - accuracy: 0.7722 - val_loss: 0.5295 -
val_accuracy: 0.7829 - lr: 6.2500e-05 - 7s/epoch - 2ms/step
Epoch 68/80
2758/2758 - 7s - loss: 0.5526 - accuracy: 0.7707 - val_loss: 0.5282 -
val_accuracy: 0.7820 - lr: 3.1250e-05 - 7s/epoch - 3ms/step
Epoch 69/80
2758/2758 - 7s - loss: 0.5521 - accuracy: 0.7696 - val_loss: 0.5297 -
val_accuracy: 0.7815 - lr: 3.1250e-05 - 7s/epoch - 2ms/step
Epoch 70/80
2758/2758 - 7s - loss: 0.5506 - accuracy: 0.7725 - val_loss: 0.5262 -
val_accuracy: 0.7835 - lr: 3.1250e-05 - 7s/epoch - 2ms/step
Epoch 71/80
2758/2758 - 7s - loss: 0.5476 - accuracy: 0.7732 - val_loss: 0.5287 -
val_accuracy: 0.7826 - lr: 3.1250e-05 - 7s/epoch - 2ms/step
Epoch 72/80
2758/2758 - 6s - loss: 0.5510 - accuracy: 0.7709 - val_loss: 0.5276 -
val_accuracy: 0.7824 - lr: 3.1250e-05 - 6s/epoch - 2ms/step
Epoch 73/80

Epoch 73: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
2758/2758 - 7s - loss: 0.5487 - accuracy: 0.7732 - val_loss: 0.5265 -
val_accuracy: 0.7830 - lr: 3.1250e-05 - 7s/epoch - 2ms/step
Epoch 74/80
2758/2758 - 7s - loss: 0.5475 - accuracy: 0.7728 - val_loss: 0.5271 -
val_accuracy: 0.7820 - lr: 1.5625e-05 - 7s/epoch - 2ms/step
Epoch 75/80
```

```
2758/2758 - 7s - loss: 0.5470 - accuracy: 0.7730 - val_loss: 0.5258 -
val_accuracy: 0.7825 - lr: 1.5625e-05 - 7s/epoch - 3ms/step
Epoch 76/80
2758/2758 - 7s - loss: 0.5477 - accuracy: 0.7726 - val_loss: 0.5260 -
val_accuracy: 0.7824 - lr: 1.5625e-05 - 7s/epoch - 2ms/step
Epoch 77/80
2758/2758 - 7s - loss: 0.5465 - accuracy: 0.7738 - val_loss: 0.5263 -
val_accuracy: 0.7821 - lr: 1.5625e-05 - 7s/epoch - 2ms/step
Epoch 78/80

Epoch 78: ReduceLROnPlateau reducing learning rate to 1e-05.
2758/2758 - 7s - loss: 0.5487 - accuracy: 0.7733 - val_loss: 0.5342 -
val_accuracy: 0.7802 - lr: 1.5625e-05 - 7s/epoch - 2ms/step
Epoch 79/80
2758/2758 - 7s - loss: 0.5453 - accuracy: 0.7746 - val_loss: 0.5265 -
val_accuracy: 0.7811 - lr: 1.0000e-05 - 7s/epoch - 2ms/step
Epoch 80/80
2758/2758 - 7s - loss: 0.5446 - accuracy: 0.7749 - val_loss: 0.5262 -
val_accuracy: 0.7811 - lr: 1.0000e-05 - 7s/epoch - 2ms/step
```

[31]:
```
plt.figure(figsize=(12, 4))

# Plot accuracy history
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Plot loss history
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```



[32]:
```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Loss: {loss}, Accuracy: {accuracy * 100:.4f}%")
```

```
811/811 [==============================] - 1s 952us/step - loss: 0.5162 -
accuracy: 0.7849
Loss: 0.5161770582199097, Accuracy: 78.4864%
```

[33]:
```
y_pred = model.predict(X_test)
y_pred = np.argmax(y_pred, axis=1)
y_test = np.argmax(y_test, axis=1)
acc = accuracy_score(y_test, y_pred)
print(f"Accuracy: {acc * 100:.2f}%")
```

```
811/811 [==============================] - 1s 782us/step
Accuracy: 78.49%
```

[34]:
```
print(classification_report(y_test, y_pred, target_names=EVENTS_AASM_NEU))
```

```
              precision    recall  f1-score   support

         REM       0.85      0.84      0.84      3175
      NREM 1       0.77      0.75      0.76      8431
      NREM 2       0.71      0.73      0.72      6063
      NREM 3       0.83      0.81      0.82      4007
        Wake       0.83      0.88      0.86      4275

    accuracy                           0.78     25951
   macro avg       0.80      0.80      0.80     25951
weighted avg       0.78      0.78      0.78     25951
```

[35]:
```
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,␣
 ↪display_labels=EVENTS_AASM_NEU.keys())
fig, ax = plt.subplots(figsize=(6, 6))
```

```
disp.plot(cmap=plt.cm.Blues,ax=ax, values_format='d', xticks_rotation=90)
ax.grid(False)
plt.show()
```