CNN

Imports

```
import keras
import mne
import numpy as np
import os
import re
import glob
from bs4 import BeautifulSoup
from keras import Sequential
from keras.src.callbacks import EarlyStopping, ReduceLROnPlateau
from keras.src.layers import Conv1D, MaxPooling1D, Dropout, Flatten,
Dense, BatchNormalization, LSTM, InputLayer, \
    GlobalAveragePooling1D, AveragePooling1D, Activation
from keras.src.utils import to_categorical
from matplotlib import pyplot as plt
from mne.preprocessing import ICA
from sklearn.model_selection import train_test_split
from tensorflow.keras.regularizers import l2
from sklearn.metrics import classification_report, accuracy_score,
ConfusionMatrixDisplay, confusion_matrix
```

Ordner mit edf-Dateien suchen und vorbereiten

```
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
```

EDF-Dateien mit gewünschter Konfig. auslesen

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

    include = [
```

```
        'EEG C4-A1',
        #'EEG C3-A2',
        #'EEG A1-A2'
        #'EOG ROC-A1',
        #'EOG LOC-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

```
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

```
tmax = 30 - 1 / 100 # 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 for 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)


    #----------------- Preprocessing ---------------------
    # ---> Butterworth Filter 0.5 - 49.5 Hz
    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'
    )
```

```
    #picks = mne.pick_types(raw_data.info, eeg=True, eog=True,
ecg=False)

    # ---> PCA (Principal Component Analysis)
    # get_data() -> transpose -> fit_transform() -> transpose back ->
make new raw or override old
    #pca_daten = raw_data.get_data(picks=picks)
    #pca = PCA()
    #pca_daten = pca.fit_transform(pca_daten.T).T


    # ---> ICA (Independent Component Analysis)
    #ica = ICA(verbose='error')
    #ica.fit(raw_data, picks=picks, 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',
        preload=True,
    )
    epochs.resample(sfreq=100, verbose='error')

    labels = epochs.events[:, 2]
```

```
    return epochs, labels
```

~~Es werden hier keine Feature extrahiert~~

Geht durch alle gefundenen Ordner und gibt fertige Epochen zurück

```
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)
        all_epochs.append(epochs)
        all_labels.append(labels)

    combined_epochs = mne.concatenate_epochs(all_epochs,
verbose='error')
    combined_labels = np.concatenate(all_labels, axis=0) if all_labels
else None

    return combined_epochs, combined_labels


main_directory = '/Volumes/Jonas_SSD/test'
X, y = process_all_folders(main_directory)
```

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

```
Data for Directory:   /Volumes/Jonas_SSD/test/00000709-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000070-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000087-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000710-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000079-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000080-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000719-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000086-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000089-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000726-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000091-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000094-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000727-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000106-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000107-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000728-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000108-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000110-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000731-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000111-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000113-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000736-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000114-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000115-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000743-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000116-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000117-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000744-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000123-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000124-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000751-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000126-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000127-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000752-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000133-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000136-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000753-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000137-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000139-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000755-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000145-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000147-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000759-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000152-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000153-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000767-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000160-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000161-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000769-A5BS00755
```

```
Data for Directory:   /Volumes/Jonas_SSD/test/00000169-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000180-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000770-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000181-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000193-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000772-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000199-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000204-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000774-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000206-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000207-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000211-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000213-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000216-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000223-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000231-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000233-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000242-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000243-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000245-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000246-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000248-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000263-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000273-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000283-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000284-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000285-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000287-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000293-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000297-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000299-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000300-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000319-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000321-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000322-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000330-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000331-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000342-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000351-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000352-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000357-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000359-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000368-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000369-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000378-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000380-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000384-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000385-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000386-A5BS00755
```

```
Data for Directory:   /Volumes/Jonas_SSD/test/00000395-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000396-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000403-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000416-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000423-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000424-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000425-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000427-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000431-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000432-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000441-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000448-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000458-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000466-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000480-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000493-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000494-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000496-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000500-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000501-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000509-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000512-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000515-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000516-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000517-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000519-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000523-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000527-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000529-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000531-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000534-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000536-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000538-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000542-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000543-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000544-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000546-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000548-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000550-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000551-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000552-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000553-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000561-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000562-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000574-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000576-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000578-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000586-A5BS00755
Data for Directory:   /Volumes/Jonas_SSD/test/00000597-A5BS00755
```

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

```
X.get_data(verbose='error').shape, y.shape
```

```
((129754, 1, 3000), (129754,))
```

```
# Example before normalization
X = X.get_data(verbose='error')
X[25, 0, :15]
```

```
array([ 3.90447318e-06, -3.23987685e-06, -1.15264144e-05, -
7.52694989e-06,
       -8.77475472e-06, -5.36156945e-06,  2.77552056e-06,
9.12160145e-06,
        3.76980608e-06,  1.45671535e-07, -1.68609282e-06, -
4.13663167e-06,
       -4.20777865e-06, -5.48460450e-06,  2.65659406e-06])
```

```
# Normalization
mean = np.mean(X, axis=2, keepdims=True)
std = np.std(X, axis=2, keepdims=True)
X = (X - mean) / std
X.shape
```

```
(129754, 1, 3000)
```

```
# Same example after normalization
X[25, 0, :15]
```

```
array([ 0.4965915 , -0.41038773, -1.4623683 , -0.95463405, -
1.11304357,
       -0.67973779,  0.35327034,  1.15890886,  0.47949544,
0.01940953,
       -0.21313397, -0.52423125, -0.5332634 , -0.69535716,
0.33817255])
```

```
# For Input_Layer
n_samples_per_epoch = X.shape[2]
n_channels = X.shape[1]
```

One-Hot Encoding

```
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]
```

```
# Reshape the data to match the expected input for 1D CNN: (n_samples,
n_samples_per_epoch, n_channels)
print("Original data shape:", X.shape)
X = np.transpose(X, (0, 2, 1))
print("Reshaped data shape:", X.shape)
```

```
Original data shape: (129754, 1, 3000)
Reshaped data shape: (129754, 3000, 1)
```

Splitting data in train and test set

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Setting parameters for early stopping and learning rate reduction

```
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5,
patience=3, min_lr=0.00001, verbose=1)
early_stopping = EarlyStopping(monitor='val_loss', patience=4,
verbose=1, restore_best_weights=True)
```

Train Models

1D-CNN Model Architektur

A Deep Learning Model for Automated Sleep Stages Classification Using PSG Signals

```
model = Sequential()
model.add(Conv1D(
    filters=64,
    kernel_size=5,
    activation='relu',
```

```
    strides=3,
    input_shape=(n_samples_per_epoch, n_channels)
))
model.add(Conv1D(filters=128, kernel_size=5,strides=2,
activation='relu'))
model.add(MaxPooling1D(pool_size=2, strides=2))
model.add(Dropout(rate=0.2))

model.add(Conv1D(filters=128, kernel_size=13, activation='relu'))
model.add(Dropout(rate=0.3))
model.add(Conv1D(filters=256, kernel_size=7, activation='relu' ))
model.add(MaxPooling1D(pool_size=2, strides=2))

model.add(Dropout(rate=0.2))

model.add(Conv1D(filters=256, kernel_size=7, activation='relu'))
model.add(Dropout(rate=0.3))
model.add(Conv1D(filters=64, kernel_size=4, activation='relu'))
model.add(MaxPooling1D(pool_size=2, strides=2))

model.add(Dropout(rate=0.2))

model.add(Conv1D(filters=32, kernel_size=3, activation='relu'))
model.add(Conv1D(filters=64, kernel_size=4, activation='relu'))
model.add(MaxPooling1D(pool_size=2, strides=2))

model.add(Dropout(rate=0.2))

model.add(Conv1D(filters=8, kernel_size=5, activation='relu'))
model.add(Conv1D(filters=8, kernel_size=2, activation='relu'))
model.add(MaxPooling1D(pool_size=2, strides=2))

model.add(Flatten())

model.add(Dense(units=64, activation='relu'))
model.add(Dense(units=5, activation='softmax'))

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

history = model.fit(X_train, y_train, epochs=50, validation_split=0.2,
callbacks=[reduce_lr, early_stopping], verbose=1)

Epoch 1/50
2596/2596 [==============================] - 721s 277ms/step - loss:
1.1331 - accuracy: 0.5198 - val_loss: 1.0002 - val_accuracy: 0.5929 -
lr: 0.0010
Epoch 2/50
2596/2596 [==============================] - 835s 322ms/step - loss:
0.9940 - accuracy: 0.5855 - val_loss: 0.9436 - val_accuracy: 0.6131 -
lr: 0.0010
```

```
Epoch 3/50
2596/2596 [==============================] - 1808s 697ms/step - loss:
0.9574 - accuracy: 0.6038 - val_loss: 0.9243 - val_accuracy: 0.6183 -
lr: 0.0010
Epoch 4/50
2596/2596 [==============================] - 1359s 523ms/step - loss:
0.9364 - accuracy: 0.6130 - val_loss: 0.9594 - val_accuracy: 0.6025 -
lr: 0.0010
Epoch 5/50
2596/2596 [==============================] - 838s 323ms/step - loss:
0.9202 - accuracy: 0.6198 - val_loss: 0.9271 - val_accuracy: 0.6248 -
lr: 0.0010
Epoch 6/50
2596/2596 [==============================] - 805s 310ms/step - loss:
0.9132 - accuracy: 0.6243 - val_loss: 0.9096 - val_accuracy: 0.6409 -
lr: 0.0010
Epoch 7/50
2596/2596 [==============================] - 814s 313ms/step - loss:
0.9027 - accuracy: 0.6311 - val_loss: 0.9232 - val_accuracy: 0.6228 -
lr: 0.0010
Epoch 8/50
2596/2596 [==============================] - 785s 302ms/step - loss:
0.8983 - accuracy: 0.6312 - val_loss: 0.9034 - val_accuracy: 0.6345 -
lr: 0.0010
Epoch 9/50
2596/2596 [==============================] - 787s 303ms/step - loss:
0.8950 - accuracy: 0.6340 - val_loss: 0.9479 - val_accuracy: 0.6162 -
lr: 0.0010
Epoch 10/50
2596/2596 [==============================] - 972s 374ms/step - loss:
0.8929 - accuracy: 0.6348 - val_loss: 0.8923 - val_accuracy: 0.6339 -
lr: 0.0010
Epoch 11/50
2596/2596 [==============================] - 771s 297ms/step - loss:
0.8901 - accuracy: 0.6353 - val_loss: 1.0149 - val_accuracy: 0.5804 -
lr: 0.0010
Epoch 12/50
2596/2596 [==============================] - 816s 314ms/step - loss:
0.8909 - accuracy: 0.6349 - val_loss: 0.8920 - val_accuracy: 0.6437 -
lr: 0.0010
Epoch 13/50
2596/2596 [==============================] - 762s 294ms/step - loss:
0.8879 - accuracy: 0.6367 - val_loss: 0.8852 - val_accuracy: 0.6404 -
lr: 0.0010
Epoch 14/50
2596/2596 [==============================] - 795s 306ms/step - loss:
0.8862 - accuracy: 0.6379 - val_loss: 0.9075 - val_accuracy: 0.6333 -
lr: 0.0010
Epoch 15/50
```

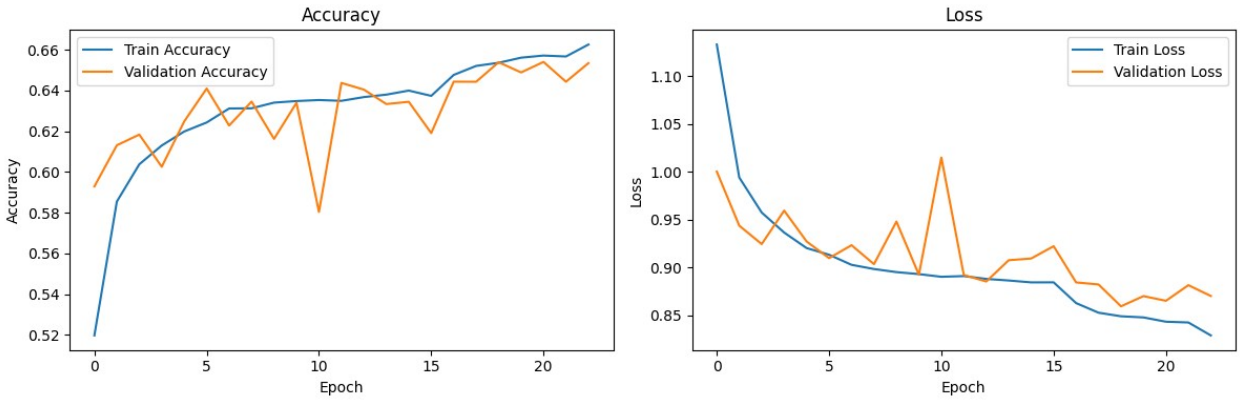
```
2596/2596 [==============================] - 820s 316ms/step - loss:
0.8843 - accuracy: 0.6399 - val_loss: 0.9092 - val_accuracy: 0.6344 -
lr: 0.0010
Epoch 16/50
2595/2596 [==============================>.] - ETA: 0s - loss: 0.8843 -
accuracy: 0.6373
Epoch 16: ReduceLROnPlateau reducing learning rate to
0.0005000000237487257.
2596/2596 [==============================] - 825s 318ms/step - loss:
0.8843 - accuracy: 0.6373 - val_loss: 0.9221 - val_accuracy: 0.6190 -
lr: 0.0010
Epoch 17/50
2596/2596 [==============================] - 806s 311ms/step - loss:
0.8626 - accuracy: 0.6476 - val_loss: 0.8842 - val_accuracy: 0.6443 -
lr: 5.0000e-04
Epoch 18/50
2596/2596 [==============================] - 811s 312ms/step - loss:
0.8525 - accuracy: 0.6520 - val_loss: 0.8821 - val_accuracy: 0.6442 -
lr: 5.0000e-04
Epoch 19/50
2596/2596 [==============================] - 484s 187ms/step - loss:
0.8488 - accuracy: 0.6535 - val_loss: 0.8592 - val_accuracy: 0.6538 -
lr: 5.0000e-04
Epoch 20/50
2596/2596 [==============================] - 842s 325ms/step - loss:
0.8476 - accuracy: 0.6561 - val_loss: 0.8699 - val_accuracy: 0.6488 -
lr: 5.0000e-04
Epoch 21/50
2596/2596 [==============================] - 545s 210ms/step - loss:
0.8431 - accuracy: 0.6571 - val_loss: 0.8650 - val_accuracy: 0.6540 -
lr: 5.0000e-04
Epoch 22/50
2595/2596 [==============================>.] - ETA: 0s - loss: 0.8423 -
accuracy: 0.6566
Epoch 22: ReduceLROnPlateau reducing learning rate to
0.0002500000118743628.
2596/2596 [==============================] - 646s 249ms/step - loss:
0.8423 - accuracy: 0.6566 - val_loss: 0.8813 - val_accuracy: 0.6442 -
lr: 5.0000e-04
Epoch 23/50
2595/2596 [==============================>.] - ETA: 0s - loss: 0.8287 -
accuracy: 0.6625Restoring model weights from the end of the best
epoch: 19.
2596/2596 [==============================] - 821s 316ms/step - loss:
0.8288 - accuracy: 0.6625 - val_loss: 0.8700 - val_accuracy: 0.6534 -
lr: 2.5000e-04
Epoch 23: early stopping
```

```
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()
```




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

```
811/811 [==============================] - 71s 87ms/step - loss:
0.8643 - accuracy: 0.6481
Loss: 0.8642942905426025, Accuracy: 64.8145%
```

```
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 [==============================] - 70s 86ms/step
Accuracy: 64.81%
```

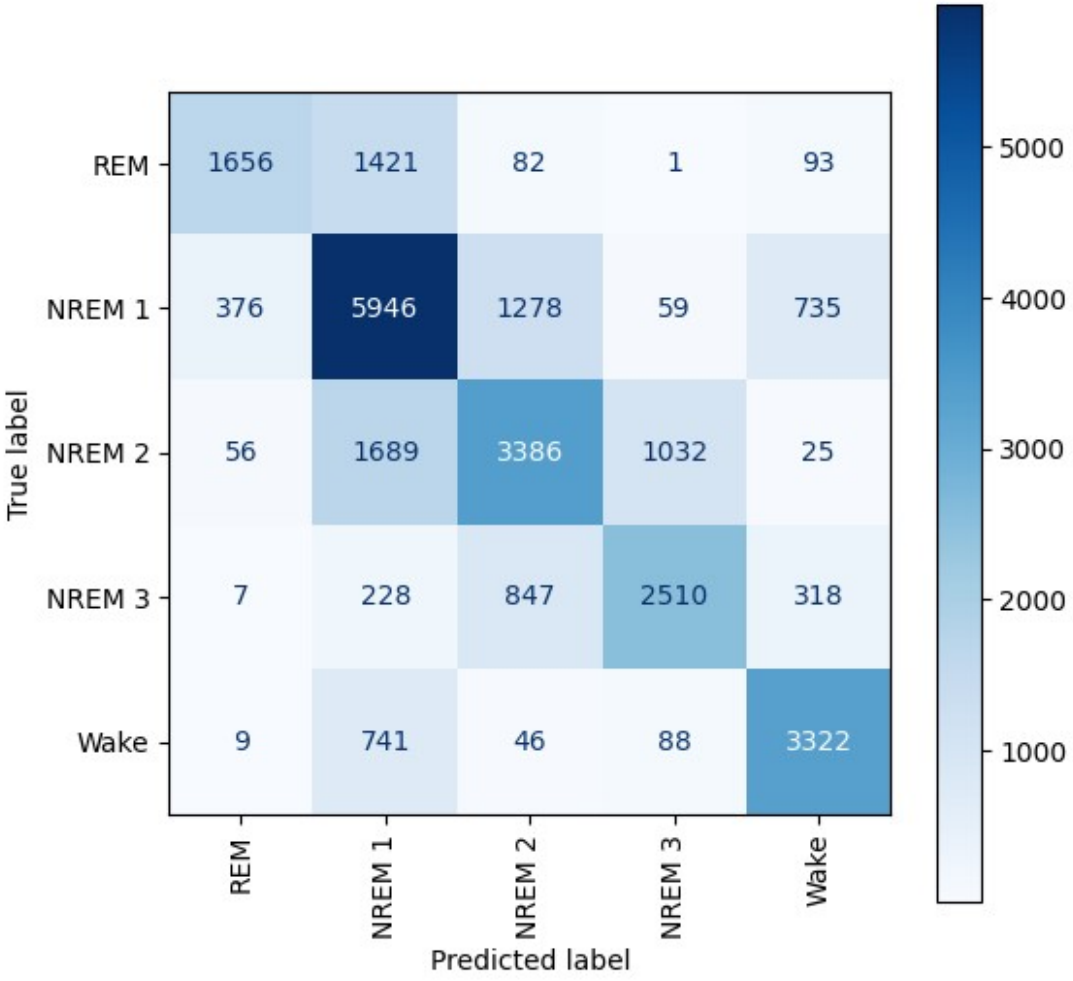
```
print(classification_report(y_test, y_pred,
target_names=EVENTS_AASM_NEU))
```

```
              precision    recall  f1-score   support

         REM       0.79      0.51      0.62      3253
      NREM 1       0.59      0.71      0.65      8394
      NREM 2       0.60      0.55      0.57      6188
      NREM 3       0.68      0.64      0.66      3910
        Wake       0.74      0.79      0.76      4206

    accuracy                           0.65     25951
   macro avg       0.68      0.64      0.65     25951
weighted avg       0.66      0.65      0.65     25951
```

```
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()
```


REM
NREM 1
NREM 2
NREM 3
Wake
True label
1656
1421
82
1
93
376
5946
1278
59
735
56
1689
3386
1032
25
7
228
847
2510
318
9
741
46
88
3322
REM
NREM 1
NREM 2
NREM 3
Wake
Predicted label
5000
4000
3000
2000
1000