Detecting Sleep States from Accelerometer Data using Deep Learning

Alvási fázisok érzékelése gyorsulásmérő adatok alapján mélytanulás használatával

Ádám M. Biró
biro.adam@edu.bme.hu

Bálint Szalay
balint.szalay@edu.bme.hu

Zsombor Szenyán
zsomborszenyan@edu.bme.hu

Budapest University of Technology and Economics
Budapest, Hungary

Abstract

Monitoring sleep patterns is essential when researching sleep and sleep-related problems. Automatically analyzing wrist-worn accelerometer data could emerge as a low-cost and practical method for logging sleep periods when conducting large-scale studies. However, traditional rule-based methods have limited accuracy in detecting sleep onset and wakeup events. In this paper we explore ways how deep learning, a powerful machine learning technique could be used to detect sleep states more precisely. We demonstrate two possible approaches to solve this problem and compare the results.

Az alvási minták monitorozása elengedhetetlen fontosságú az alvással kapcsolatos kutatások során. A csuklón hordható gyorsulásmérő adatok automatikus elemzése egy olcsón és egyszerű megoldás lehet az alvási ciklusok rögzítésére nagy mintájú kutatások során. Azonban a hagyományos szabályalapú módszerek nem tudják megfelelő pontossággal meghatározni az elalvás és felébredés időpontját. Ebben a dokumentumban megvizsgálunk lehetséges módszereket arra, hogy a mélytanulást, a gépi tanulás egyik formáját használjuk fel az alvási állapot pontosabb meghatározására. Megmutatunk két lehetséges megközelítést, és összevetjük az eredményeket.

1 Introduction

Fulfilling sleep plays a crucial role in maintaining physical and mental well–being. Yet a great number of people suffer from problems which negatively affect their sleep patterns. Inadequate quality and quantity of sleep is associated with various health problems, including depression and cardiovascular diseases.

Monitoring sleep patterns is essential when researchers conduct studies on sleep. However, detecting the onset and end of sleep phases is a challenging task. Invasive sleep monitoring methods, such as polysomnography (PSG) are expensive and unsuitable for large–scale studies.

Preprint. Under review.

In recent years, wearable devices equipped with accelerometers have emerged as a possible tool for non–invasive sleep monitoring. Traditional techniques which use human–engineered features achieve moderate success in recognizing sleep onset and wakeup events from accelerometer data due to the nuanced transition between sleep and awake states.

A Kaggle competition was hosted by the Child Mind Institute [4] with the goal of finding new approaches to sleep state detection from accelerometer data utilizing the latest advancements in the field of data science.

In this paper we explore the ways how deep learning, a promising machine learning method could be used to solve the task, and also document our models which we created as an entry to the Kaggle competition. Our work also serves as a homework project for the course Deep Learning in Practice at Budapest University of Technology and Economics.[1]

2 Dataset

For our model, we used the dataset provided in the Kaggle competition [4]. The training dataset contained about 270 recordings of wrist-worn accelerometer data, annotated with two event types: onset, the beginning of sleep and wake up, the end of a sleep phase.

The accelerometer recordings are time series data which contain two features: the *z-angle* and *ENMO*. The z-angle is a metric derived from individual accelerometer components that is commonly used in sleep detection, and refers to the angle of the arm relative to the vertical axis of the body. ENMO is the Euclidean Norm Minus One of all accelerometer signals, with negative values rounded to zero. The time series have a resolution of five seconds intervals between data points.

The data was labeled by sleep experts. The following policy was used to label sleep events according to the Kaggle competition description [2]:

> "While sleep logbooks remain the gold-standard, when working with accelerometer data we refer to sleep as the longest single period of inactivity while the watch is being worn. For this data, we have guided raters with several concrete instructions:
>
> - A single sleep period must be at least 30 minutes in length A single sleep period can be interrupted by bouts of activity that do not exceed 30 consecutive minutes
> - No sleep windows can be detected unless the watch is deemed to be worn for the duration (elaborated upon, below) The longest sleep window during the night is the only one which is recorded
> - If no valid sleep window is identifiable, neither an onset nor a wakeup event is recorded for that night.
> - Sleep events do not need to straddle the day-line, and therefore there is no hard rule defining how many may occur within a given period. However, no more than one window should be assigned per night. For example, it is valid for an individual to have a sleep window from 01h00-06h00 and 19h00-23h30 in the same calendar day, though assigned to consecutive nights
> - There are roughly as many nights recorded for a series as there are 24-hour periods in that series."

Figure 2. shows an example plot from the input data, a portion of a time series labeled with sleep states.

3 Exploring possible methods

We began our work by exploring existing work related to the subject. Loh et al. [6] and Malafeev et al. [7] showed that deep neural networks can be utilized for classification of sleep stages based

[1] Deep Learning in Practice with Python and LUA. `http://smartlab.tmit.bme.hu/oktatas-deep-learning`

[2] `https://www.kaggle.com/competitions/child-mind-institute-detect-sleep-states/data`

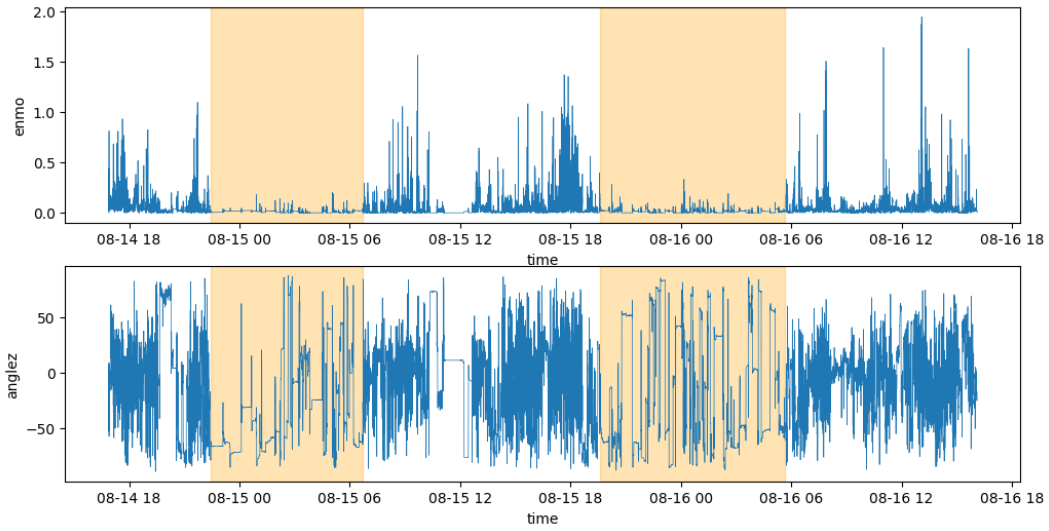

Figure 1: A roughly two–day period of a time series data, annotated with sleep states. The upper diagram shows the ENMO signal and the bottom diagram shows the Z–angle signal. The highlighted intervals are labeled as sleep periods.

on brainwave signal time series, such as polysomnography (PSG) recordings. Roberts et al. [8] showed that motion sensor data from consumer-grade wearables could possibly be used for sleep stage classification, as there is correlation between motion sensor data and PSG signal data. This led us to believe that deep learning could be successfully used on accelerometer data to detect sleep events. However, the previous works focused on classifying the sleep stage based on an fixed-length window of time series sequences, rather than providing a method to accurately find the boundaries of these stages in an series of arbitrary length.

Typically, the output of a neural network is of a fixed-length. But in our case, the number of sleep onset and wakeup is variable, so we had to design a solution that can produce variable-length output. A straightforward approach would be to divide the time series into smaller segments, as there is at most one sleep period recorded per night. However, this is problematic because there is no hard definition on the boundaries of a night, and the length of a sleep period can vary significantly.

We tried two different approaches to solve this problem: a change point detection based approach and a momentary sleep state classification approach. Below we present the basic principle of these two approaches.

4 Approach 1: Change point detection (CPD)

4.1 Overview

Change points are points in a time series that mark the boundaries of certain states. In our problem, sleep and awake phases can be viewed as states and the change points represent onset and wake up events. Various methods exist to detect change points using classical machine learning. See Aminikhanghahi et al. [1] for a survey of possible methods. Some works have examined the possibility of using deep neural networks for change point detection, notably Li et al. [5] and Ebrahimzadeh et al. [3]. We used an approach similar to those that are outlined in these works.

Firstly, we trained a classifier deep neural network to recognize if there is a change point in a given segment of the sensor data series. The input is a fixed-length segment of the sensor data, which we call a *frame*. The output is a one-hot encoded representation of either one of the following three classes: *no event*, *onset*, *wake up*. Then we used a sliding window technique to detect onset and wake up events during the prediction. We employed simple algorithmic methods to merge events that were detected multiple times and filter out sleep periods that are too short or should not be recorded according to the policy described in section 2.

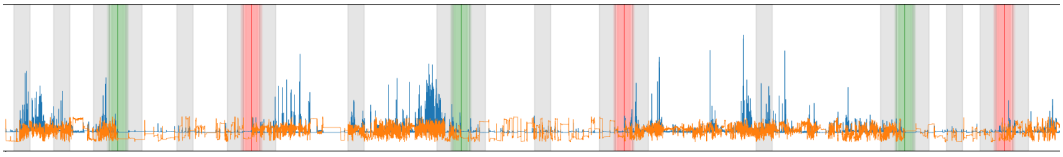

Figure 2: A segment of a training sensor series marked with the sampled frames. Grey, green and red frames show sampled frames labeled as *no event*, *onset*, and *wake up*, respectively.

We further refer to this approach as *CPD*.

4.2 Data preprocessing

We used two features as an input from the time series: ENMO and Z-angle. We standardize the two signals to have zero mean and one standard deviation. Then we sampled frames around ground-truth onset events, wake up events and in between events with the appropriate labels. Figure 2. shows a segment of a training series marked with sampled and labeled frames. Figure 3. shows example frames that were assigned to each classes. The frame length was set to 60 minutes. Longer frame lengths improved the classification performance, but made the localization of events less accurate as many frames overlapped with one event.

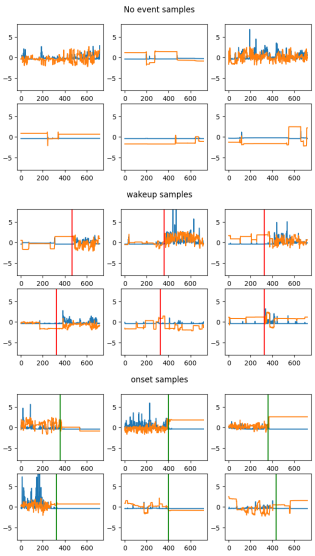

Figure 3: Training samples in each class extracted from the training data.

4.3 Model architecture

We used a multilayered artificial neural network with one-dimensional convolutional layers. Convolutional layers are ideal for extracting features in an input which is indifferent to translation. The model consists of three convolutional layers with additional subsampling layers, and three fully-connected layers. The output layer has softmax activation because the output is an one-hot encoded representation of the predicted class. We used regularization to counter overfitting. The detailed architecture of the model can be seen on figure 4.3. We also tried a recurrent architecture, but there was no improvement in the classification performance.

4.4 Training

For the training we used the Adam optimization algorithm and categorical crossentropy as a loss function. The training dataset contained around 25.000 samples in each class. The best achieved validation loss was 0.18 and with 94% classification accuracy.

4.5 Prediction

We used the following algorithm to make predictions using the trained classifier model:

1. Slide a frame-sized window with a stride of 3 minutes. Feed the frame at each window position to the classifier network. If the predicted probability of an onset or wake up event exceeds a certain threshold, record the event accordingly at the center of the window. We got the best results with 0.97 as a threshold.
2. Combine multiple events of the same type into one event if they are close to each other, which means they probably represent the same ground-truth event. The time step of the new combined event is the average of the combined events.

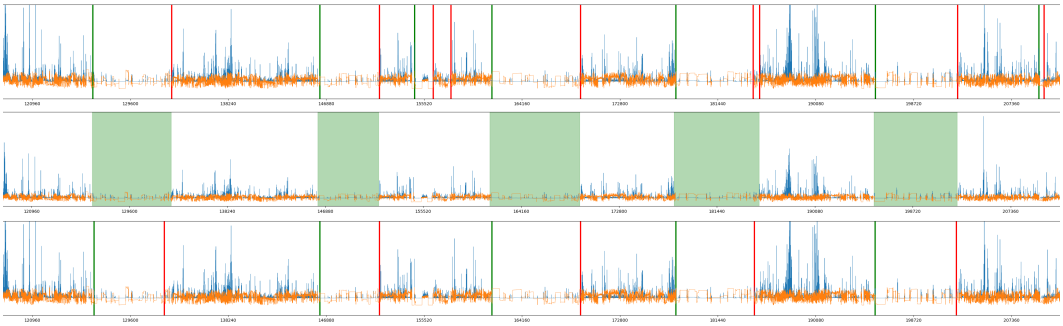

Figure 5: An example of result of the prediction algorithm. The uppermost plot shows the combined predicted events, the middle plot shows the fitted and filtered sleep periods. The bottom plot shows the ground truth events. Red lines represent onset events, green lines represent wake up events. The ENMO and Z-angle signals are colored orange and blue, respectively.

3. Determine the sleep periods based on the predicted events with by choosing the longest possible sleep periods. This effectively means sleep periods start when there is a predicted onset before a wakeup event and ends when there is a wakeup event before an onset event.
4. Merge sleep periods which have a less than 30 minute long gap between them. This is useful because according to the instructions used in recording the ground-truth events, sleep periods can be interrupted with less than 30-minute long bouts of activity (see section 2).
5. Filter out sleep periods that are shorter than 30 minutes (which are not recorded according to the instructions).
6. Assign each predicted sleep period to a night based on which *midnight point* it its midpoint is the closest to. Midnight point is a designated time of the day, which we set to 3AM.
7. Take only the longest sleep period for each night. This is needed because the rest are not recorded according to the instructions. Furthermore this helps to filter out inactive periods during the day which are not sleeping states.

See figure 5. for a visual example on the algorithm.

4.6 Overview

The other method is based on predicting the asleep/awake state in each time step based on a fixed-length segment of sensor data. In each case, we try to predict whether a person was awake or asleep at each step. There are several attempts for this approach notably Sano et al. [9] and Chen et al. [2]. We tried to make a classifier based on these approaches.

Firstly, we trained a deep neural network using BiLSTM (see Zargar et al. [10]) to recognize if a person was awake or was asleep at each step in a given segment of the sensor data series. The output represents the probability of a person being awake at each step. Then we used a simple algorithm like the one mentioned in 4.5.

We further refer to this approach as *MSC*.

4.7 Data preprocessing

We used one feature as an input from the time series: ENMO. We did not use Z-angle because the model's performance was worse when both ENMO and Z-angle were used. In addition we used two derived features from ENMO. The first was a moving average where we took the average of 7 consecutive values of ENMO. The second was a shifted value, where we took the previous value of the ENMO at the given timestamp. We used min-max scaling for standardization. Then we sampled frames around ground-truth onset events, wake up events and in between events. The frame length was set to 200 timestamp.

4.8 Model architecture

We used a multilayered artificial neural network with bidirectional LSTM layers. BiLSTM layers are well-suited for processing sequential data, allowing us to consider context both forward and backward at each step. After that we added a fully-connected layer with sigmoid activation to predict the probability of sleeping. If the probability is closer to 1, then the person was asleep otherwise was awake. We did not use any regularization method, because the model still had difficulty to learn. The detailed architecture of the model can be seen on figure 4.8.

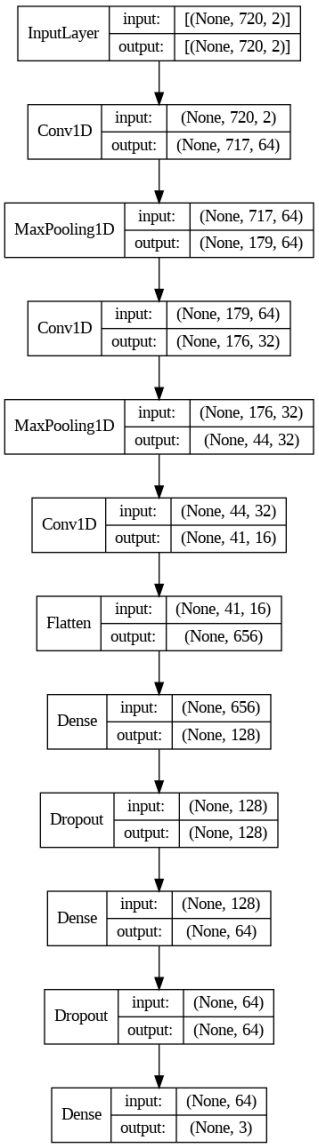

Figure 4: An overview of the neural network of the CPD approach.

4.9 Training

For the training we used the Adam optimization algorithm and binary crossentropy as a loss function. The training dataset contained around 4.000 samples in each class. The best achieved validation loss was 0.15.

4.10 Prediction

We used the following algorithm to make predictions using the trained classifier model:

1. First of all, we had predicted values for each steps. In order to filter out possible anomalies, we took a 240-step long moving average for the probabilities.
2. After that if the probabilities is higher than 0.5 then at that step the person was asleep so the value of sleeping was 1. Otherwise the people was awake so the value of sleeping was 0. We extracted onset and wakeup events from these values.

3-6. Same as in the CPD approach.

5 Evaluation

5.1 Metric

For the evaluation of the two models, we used a metric called Event Detection Average Precision. This metric is used by Kaggle to score the submissions to the competition. The metric is described on Kaggle as the following:[3]

> This metric is similar to IOU-threshold average precision metrics commonly used in object detection. For events occuring in time series, we replace the IOU threshold with a time tolerance.
>
> Submissions are evaluated on the average precision of detected events, averaged over timestamp error tolerance thresholds, averaged over event classes.
>
> Detections are matched to ground-truth events within error tolerances, with ambiguities resolved in order of decreasing confidence.

[3]Source: Event Detection AP | Kaggle. `https://kaggle.com/code/metric/event-detection-ap`

We first evaluated the models on 30 randomly series from the original Kaggle training dataset which were not used during the training. Then we submitted the models to the Kaggle competition, which were scored with a private testing dataset. The threshold values used are 1, 3, 5, 7.5, 10, 12.5, 15, 20, 25, 30 minutes as specified by Kaggle.

5.2 Results

The two approaches achieved the following scores in the Kaggle competition:

CPD model best score: **0.367**
MSC model best score: **0.317**

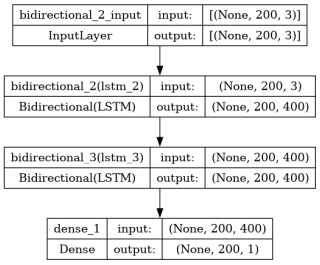

Figure 6: An overview of the neural network of the momentary classification approach.

6 Observations

Both models performed similarly on the testing data. They were moderately successful in recognizing sleep patterns and scored significantly lower scores than the state-of-the-art solutions. (The competition winner submission scored 0.852.)

The CPD based model is capable of succesfully recognizing almost all ground-truth onset and wake up events, and achieves moderate precision in the localization of the events, which significantly detoriates if the frame size is increased. The greatest limiting factor of the model is that it produces many false-positive predictions, which alter the boundaries of the corresponded sleep periods. It is prone to detecting false positive events partly due to the fact that it only sees a short context of a change point. As a result, this model labels all periods with little variation in sensor data as sleep periods, including intervals where the accelerometer was removed or the subject was resting but was not asleep, which can easily be distinguished from sleep periods when the whole interval is seen.

References

[1] Samaneh Aminikhanghahi and Diane J. Cook. A survey of methods for time series change point detection. *Knowledge and Information Systems*, 51(2):339–367, 2017.

[2] Zhenghua Chen, Min Wu, Wei Cui, Chengyu Liu, and Xiaoli Li. An Attention Based CNN-LSTM Approach for Sleep-Wake Detection With Heterogeneous Sensors. *IEEE Journal of Biomedical and Health Informatics*, 25(9):3270–3277, 2021.

[3] Zahra Ebrahimzadeh, Min Zheng, Selcuk Karakas, and Samantha Kleinberg. Deep learning for multi-scale changepoint detection in multivariate time series. *CoRR*, abs/1905.06913, 2019.

[4] Nathalia Esper, Maggie Demkin, Ryan Hoolbrok, Yuki Kotani, Larissa Hunt, Andrew Leroux, Vincent van Hees, Vadim Zipunnikov, Kathleen Merikangas, Michael Milham, Alexandre Franco, and Gregory Kiar. Child Mind Institute - Detect Sleep States, 2023. URL: `https://kaggle.com/competitions/child-mind-institute-detect-sleep-states`.

[5] Jie Li, Paul Fearnhead, Piotr Fryzlewicz, and Tengyao Wang. Automatic change-point detection in time series via deep learning. *arXiv preprint arXiv:2211.03860*, 2022.

[6] Hui Wen Loh, Chui Ping Ooi, Jahmunah Vicnesh, Shu Lih Oh, Oliver Faust, Arkadiusz Gertych, and U. Rajendra Acharya. Automated detection of sleep stages using deep learning techniques: A systematic review of the last decade (2010–2020). *Applied Sciences*, 10(24), 2020.

[7] Alexander Malafeev, Dmitry Laptev, Stefan Bauer, Ximena Omlin, Aleksandra Wierzbicka, Adam Wichniak, Wojciech Jernajczyk, Robert Riener, Joachim Buhmann, and Peter Achermann. Automatic Human Sleep Stage Scoring Using Deep Neural Networks. *Frontiers in Neuroscience*, 12, 2018.

[8] Daniel M Roberts, Margeaux M Schade, Gina M Mathew, Daniel Gartenberg, and Orfeu M Buxton. Detecting sleep using heart rate and motion data from multisensor consumer-grade wearables, relative to wrist actigraphy and polysomnography. *Sleep*, 43(7):zsaa045, 03 2020.

[9] Akane Sano, Weixuan Chen, Daniel Lopez-Martinez, Sara Taylor, and Rosalind W. Picard. Multimodal Ambulatory Sleep Detection Using LSTM Recurrent Neural Networks. *IEEE Journal of Biomedical and Health Informatics*, 23(4):1607–1617, 2019.

[10] Sakib Zargar. Introduction to Sequence Learning Models: RNN, LSTM, GRU. 04 2021.