Deep residual networks for automatic sleep stage classification of raw polysomnographic waveforms

Olesen, Alexander Neergaard; Jennum, Poul; Peppard, Paul; Mignot, Emmanuel; Sørensen, Helge Bjarup Dissing

Published in:
Proceedings of 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society

Link to article, DOI:
10.1109/EMBC.2018.8513080

Publication date:
2018

Document Version
Early version, also known as pre-print

Citation (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Deep residual networks for automatic sleep stage classification of raw polysomnographic waveforms

Alexander N. Olesen¹,¹,², Poul Jennum³, Paul Peppard⁴, Emmanuel Mignot², and Helge B. D. Sorensen¹.

Abstract—We have developed an automatic sleep stage classification algorithm based on deep residual neural networks and raw polysomnogram signals. Briefly, the raw data is passed through 50 convolutional layers before subsequent classification into one of five sleep stages. Three model configurations were trained on 1850 polysomnogram recordings and subsequently tested on 230 independent recordings. Our best performing model yielded an accuracy of 84.1% and a Cohen’s kappa of 0.746, improving on previous reported results by other groups also using only raw polysomnogram data. Most errors were made on non-REM stage 1 and 3 decisions, errors likely resulting from the definition of these stages. Further testing on independent cohorts is needed to verify performance for clinical use.

I. INTRODUCTION

Sleep staging is the principal tool available to medical doctors in the analysis of sleep disorders. Natural human sleep consists of recurring cycles of three to four distinct phases, which are primarily characterized by changes in brain activity, eye movements, muscle activations and breathing. A polysomnogram (PSG) containing electroencephalography (EEG), electrooculography (EOG), electromyography (EMG) and other signals is collected during sleep, and subsequently processed and analyzed by sleep technicians according to standards by the American Academy of Sleep Medicine (AASM). Each 30 s epoch of data is categorized into either wakefulness (W), rapid eye movement (REM) sleep, or one of three stages of non-REM sleep (N1, N2, N3) [1]. However, this approach is prone to subjective interpretation of sleep staging rules, which have prompted extensive research in using various signal processing and machine learning approaches [2].

Attempts at exploiting deep learning models for sleep staging have been proposed recently. One group used a transfer learning-approach to characterize sleep stages [3], where 30 s epochs of Fpz-Cz EEG were subjected to multitaper spectral estimation (MTSE) in order to create spectral image representations [4], that ultimately were fed as input to a VGG-16 model stripped of the last layers [5]. This approach was cross-validated using a leave-one-out scheme on 20 subjects and yielded a bootstrapped accuracy of 86% ± 2%.

Using MTSE for representing EEG data was also investigated in [6], where the authors compared various machine and deep learning models trained on either raw EEG waveforms, MTSE spectrograms, or 96 expert-defined features. They tested their best performing model on recordings from 1000 individual subjects and obtained an accuracy of 85.76% and a Cohen’s kappa of 0.79 using a combination of expert-defined features and recurrent neural networks (RNN). On the same test set, they obtained accuracy/kappa values of 77.31% and 0.71 using a deep learning model trained on raw EEG waveforms.

However, it is still unclear whether manual feature extraction such as sleep spindle/K-complex detection, or data transformations, such as spectrograms or MTSE, are strictly necessary for efficient deep learning, and there is still room for improvement in the current state of the art for raw PSG analysis.

We propose a novel method for automatic sleep staging combining state of the art deep learning networks with raw PSG data to accurately capture the complex relationships found in PSG data without resorting to data transformations and manual feature engineering.

II. DATA

A database containing 2310 recordings extracted from the Wisconsin Sleep Cohort was used in this study. Specific acquisition details concerning the PSGs are described in [7]. The entire set of PSG studies was randomly split into training (train), validation (eval), and testing (test) subgroups in an 8:1:1 ratio. Detailed demographic information as well as relevant PSG variables for all three subgroups are provided in table I including apnea-hypopnea index (AHI) and time spent in each sleep stage based on manual scoring.

III. METHODS

A. Signal extraction and pre-processing

Central and occipital EEG from right hemisphere, left and right EOG, and chin EMG channels were extracted from each PSG study. To accommodate different equipment setups used for recording studies, each channel was upsampled to 200 Hz. Following resampling, signals were filtered using zero-phase Butterworth filters with frequency ranges recommended by the AASM [1]. Since dynamic ranges vary considerably across channels, each signal was soft-normalized using the 5th and 95th quantiles, such that

$$x_{\text{norm}} = 2 \frac{x - Q_{0.05}(x)}{Q_{0.95}(x) - Q_{0.05}(x)} - 1,$$  \hspace{1cm} (1)
To reduce computational costs, we introduce a singleton dimension.

\[ \text{tf.layers.conv2d} \]

input argument to subsequently call were set to 1850 (1010) 230 (120) 0.210 and generalization compared to vanilla residual blocks [9].

activation layer, and a conv layer. This pre-activation con-

ventional (conv) and dense layers arranged in four block layers in Fig. 1. Briefly, the bulk network comprised 50 convolua-

competition [8].

network. This feature allows for extremely deep network to maintain a proper gradient backpropagation through the employ residual skip connections between layers in order

number of signals, and number of timesteps for one epoch,

respectively.

\[ p \text{-value} \]

TABLE I

EXTRACTED WSC cohort demographics for each subgroup. Significant \( p \)-values are highlighted in bold.

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Eval</th>
<th>Test</th>
<th>( p )-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n ) (male)</td>
<td>1850 (1010)</td>
<td>230 (112)</td>
<td>230 (120)</td>
<td>0.210</td>
</tr>
<tr>
<td>Age (years)</td>
<td>59.2 ( \pm ) 8.4</td>
<td>59.9 ( \pm ) 8.5</td>
<td>60.4 ( \pm ) 8.2</td>
<td>0.092</td>
</tr>
<tr>
<td>BMI (kg m(^{-2}))</td>
<td>31.7 ( \pm ) 7.2</td>
<td>31.0 ( \pm ) 6.9</td>
<td>32.2 ( \pm ) 7.7</td>
<td>0.203</td>
</tr>
<tr>
<td>AHI (h(^{-1}))</td>
<td>12.6 ( \pm ) 15.6</td>
<td>11.5 ( \pm ) 14.9</td>
<td>12.4 ( \pm ) 16.2</td>
<td>0.600</td>
</tr>
<tr>
<td>PSG dur. (h)</td>
<td>7.4 ( \pm ) 0.8</td>
<td>7.4 ( \pm ) 0.7</td>
<td>7.4 ( \pm ) 0.8</td>
<td>0.947</td>
</tr>
<tr>
<td>( W ) (%)</td>
<td>18.5 ( \pm ) 11.3</td>
<td>17.2 ( \pm ) 11.1</td>
<td>19.6 ( \pm ) 11.8</td>
<td>0.071</td>
</tr>
<tr>
<td>N1 (%)</td>
<td>8.2 ( \pm ) 4.5</td>
<td>8.8 ( \pm ) 5.6</td>
<td>8.9 ( \pm ) 5.1</td>
<td>0.038</td>
</tr>
<tr>
<td>N2 (%)</td>
<td>54.2 ( \pm ) 10.3</td>
<td>54.0 ( \pm ) 10.9</td>
<td>52.4 ( \pm ) 11.0</td>
<td>0.048</td>
</tr>
<tr>
<td>N3 (%)</td>
<td>5.8 ( \pm ) 6.4</td>
<td>6.4 ( \pm ) 7.0</td>
<td>6.0 ( \pm ) 7.0</td>
<td>0.433</td>
</tr>
<tr>
<td>REM (%)</td>
<td>13.3 ( \pm ) 5.9</td>
<td>13.7 ( \pm ) 5.8</td>
<td>13.2 ( \pm ) 5.7</td>
<td>0.635</td>
</tr>
</tbody>
</table>

where \( x_{\text{norm}} \) denotes the normalized version of the signal \( x \), and \( Q_{0.05}(x) \) and \( Q_{0.95}(x) \) denotes the 5th and 95th percentile, respectively. Doubling and subtracting by one rescales \( Q_{0.05}(x) \) and \( Q_{0.95}(x) \) to \(-1\) and \(1\), respectively.

Finally, each signal was segmented into 30 s epochs corre-

sponding to AASM criteria [1], resulting in a tensor \( X \) with elements

\[ (x_{nct}) \in \mathbb{R}^{N \times C \times 1 \times T}, \]

with \( N = 16 \), \( C = 5 \), and \( T = \) 6000 being batch size, number of signals, and number of timesteps for one epoch, respectively.\(^1\)

B. Deep residual network model

We applied a deep learning model inspired by the residual network models proposed in [8], [9]. These types of models employ residual skip connections between layers in order to maintain a proper gradient backpropagation through the network. This feature allows for extremely deep network structures, and a specific variant of this model with 152 layers came in 1st place in the ILSVRC ’15 image classification competition [8].

1) Architecture: The residual network model is illustrated in Fig. 1. Briefly, the bulk network comprised 50 convolutional (conv) and dense layers arranged in four block layers of four bottlenecked residual blocks each.

A single bottleneck residual block contains three triplets of a batch normalization layer, a rectified linear unit (ReLU) activation layer, and a conv layer. This pre-activation configuration has shown benefits with regards to trainability and generalization compared to vanilla residual blocks [9]. Projection shortcuts were used between the first ReLU and conv layers to the output of the last conv layer. Kernel sizes were set to \(1 \times 1\) for the first and third conv layers, and \(1 \times 3\) for the second conv layer. The number of output filters for each residual block was \(l \times f\) with \(l\) being the block layer index and \(f = 16\), resulting in a total of 256 filters after the final conv layer.

Before the bottleneck blocks, the input tensor \( X \) was passed through an initial conv layer consisting of \(64 \times 1 \times 16\) filters, and then through a maximum pooling (max pool) layer with a \(1 \times 2\) kernel and stride size, effectively reducing the time-resolution by a factor of \(2\). This max pool operation was implemented in the beginning of each block layer.

The output tensor from the block layers was subsequently passed to a final batch normalization and ReLU activation layer, followed by a mean pooling layer to reduce the tensor to \( X = (x_{nk}) \in \mathbb{R}^{N \times 5^{16}} \). Finally, a fully connected layer with \(K = 5\) output units corresponding to the sleep stages resulted in the following output tensor

\[ \mathbf{P} = (p_{nk}) \in \mathbb{R}^{N \times K}, \quad p_{nk} = \frac{\exp z_{nk}}{\sum_{k} \exp z_{nk}} \]

with \(p_{nk}\) containing the softmax activations of the output units \(z_{nk}\) from the fully connected layer for the \(n\)th subject and the \(k\)th sleep stage. The predicted class for the \(n\)th subject can then be calculated as

\[ \hat{y}_n = \arg \max_k p_{nk}. \]

2) Training: The optimization problem was constructed using cross entropy loss across \(K\) classes and \(N\) epochs as objective function, such that

\[ \mathcal{L}(p_n | y_n, W) = - \sum_{k=1}^{K} y_{nk} \log p_{nk}, \]

\(1\)The 1-dimensional convolution \text{tf.layers.conv1d} reshapes the input argument to subsequently call \text{tf.layers.conv2d} in TensorFlow.
is the calculated cross entropy loss for epoch \( n \) given predicted class probabilities \( p_n \), true class labels \( y_n \), and the set of current weights \( W \). Then, the average cost across a batch of data is

\[
C(P|Y, W) = \frac{1}{N} \sum_{n=1}^{N} L(p_n|y_n, W).
\]

The cost function was optimized using the Adam optimization algorithm with default hyperparameters [10]. Weights were initialized using variance scaling [11], and we applied weight decay during training with \( \lambda = 10^{-4} \). The initial learning rate was set to \( \alpha = 10^{-3} \) and was multiplied by 0.1 every 50000 steps.

In order to investigate the effect of the imbalanced data on the network performance, we trained the following three different configurations. First, we defined a baseline configuration as described in the previous sections. The second was a weighted configuration, where the cost function in eq. (6) was replaced with an average weighted by the inverse frequency for the correct class, such that

\[
C(\hat{Y}|Y, W) = \frac{\sum_{n} \omega_n(y_n)C(\hat{y}_n|y_n, W)}{\sum_{n} \omega_n(y_n)},
\]

where \( \omega_n(y_n) \) is the inverse frequency for the correct class for the \( n \)th subject in the current batch. Finally, a balanced configuration was tested, in which we performed resampling of the training dataset in order to balance classes. We oversampled the N1, N3, and REM classes with replacement, while undersampling the N2 class in order to have approximately equal fractions of each class in total.

Models were implemented in TensorFlow 1.4, and trained on a single workstation running Ubuntu 16.04 with a Ryzen 7 1700X 8-core CPU, an NVIDIA GTX 1080 Ti GPU with 11 GB memory, and 32 GB RAM memory.

C. Performance metrics

Individual precision, recall and F1 scores (Pr, Re, F1) were calculated for each sleep stage and subsequently aggregated for each recording by stage frequency weighting, such that

\[
\text{Pr}_{nk} = \frac{TP}{TP + FP}, \quad \text{Pr}_n = \frac{\sum_k \beta_{nk} \text{Pr}_{nk}}{\sum_k \beta_{nk}}
\]

\[
\text{Re}_{nk} = \frac{TP}{TP + FN}, \quad \text{Re}_n = \frac{\sum_k \beta_{nk} \text{Re}_{nk}}{\sum_k \beta_{nk}}
\]

\[
F1_{nk} = 2 \cdot \frac{\text{Pr}_{nk} \cdot \text{Re}_{nk}}{\text{Pr}_{nk} + \text{Re}_{nk}}, \quad F1_n = \frac{\sum_k \beta_{nk} F1_{nk}}{\sum_k \beta_{nk}},
\]

where \( \beta_{nk} \) is the frequency of stage \( k \) for recording \( n \), and TP, FP and FN are true positives, false positives, and false negatives, respectively. Overall accuracy (Acc) and Cohen’s kappa (\( \kappa \)) were also calculated for each recording. All metrics were summarized by mean and standard deviations.

D. Statistical tests

Demographic and PSG variables were tested with ANOVAs after establishing normality, while gender was tested with a \( \chi^2 \) test. Significance was set at \( p = 0.05 \).

IV. Results and Discussion

Performance metrics for the train and eval subgroups are shown in table II. Not accounting for Pr, the baseline configuration compares favorably to the weighted and balanced configurations on both subgroups with an average accuracy of 85.0% and a Cohen’s kappa of 75.4 on the eval subgroup. Since the training data is imbalanced in favor of N2, it would be fair to assume overfitting to the majority class, however, the lower spread in both precision and recall does not support this. Evaluating the baseline model on the test subgroup gave only a slight drop in accuracy and \( \kappa \), indicating that the model generalizes well, see table III and table IV. The lowest sensitivity is obtained for N1 and N3, which is in accordance with clinical experience reported in the literature [12]–[15]. N1 is a transitional stage between wakefulness, drowsiness and sleep often containing beta and alpha activity in epochs of low interscorer agreement, which explains the low predictive power in the confusion matrix. The sleep continuum is also apparent in Fig. 2 which shows the manually and automatically scored hypnograms in the middle and bottom traces, and the hypnodensity graph in the top trace for a representative subject in the test subgroup. The hypnodensity is a probabilistic representation of the hypnogram, which has found use in the detection of Parkinson’s and narcolepsy [16]–[18]. Our baseline model attains favorable performance when comparing to the results reported for the raw waveform CNN model in [6] with both higher accuracy and Cohen’s kappa. However, it should be
stressed that [6] used EEG from 9000 recordings, while our model uses EEG, EOG and EMG from 1850 recordings. Furthermore, our baseline model performs only slightly worse compared to the best-performing model using manual feature engineering and RNNs in [6]. This indicates a possible performance gain by adding recurrent networks, such as long short-term memory cells, to our network.

A possible limiting factor to our model is the filter kernels. The small filter sizes in block layers might not be able to accurately capture the physiological dynamics, but there are indications that many, smaller kernels are preferable to fewer, larger kernels when comparing model complexity versus computational costs [19].

Future work will include adding more data to balance classes, and adding long short-term memory cells to the network in order to model temporal dynamics between epochs. As we performed minimal hyperparameter tuning in this work, investigating the effects of changing the network specifications to optimize performance is also a relevant area of future research.

V. CONCLUSION

We have shown that common data transformations such as spectrograms are not necessary for automatic sleep staging. Combining residual learning networks and raw PSG waveforms, we obtained an average accuracy of 84.1% and Cohen’s kappa of 0.745, improving on previously reported results on raw PSG sleep staging. Further testing on independent cohorts will illuminate the clinical applicability of this method, while introducing more data and memory cells will be explored to increase performance even further.

REFERENCES