Learning Sleep Stages

https://www.youtube.com/watch?v=njH4z7iWECM&t=208s

Ria Verma, MS[1], Ayush Baid, MS[1], Yu Jia Shai Xie, MS[1]
[1]Georgia Institute of Technology, Atlanta, GA

Abstract

Sleep quality and sleep disorder have shown high correlations with various health problems such as cardiovascular disease [1], homeostasis imbalance [2], and even Alzheimer's [3]. With the rise of wearable devices, an increasing number of people are interested in their sleep quality. Sleep stage scoring is traditionally done manually by sleep experts who look at EEG signals produced during sleep. EEG signals are a common input for sleep scoring, and the manual process to label the sleep stages is slow. We propose a novel deep learning architecture on a single EEG channel using CNNs to extract features and Transformers to leverage neighboring temporal context to predict sleep stage labels. We achieve an accuracy of 75% and a mean-F1 score of 75.05% on the Physionet-2018 dataset, which is a marginal improvement in F1-scores over other similar methods.

Introduction

Sleep quality is a major concern for sleep-related disorders, and sleep stages can be representative to the sleep behaviors, our group has determined to analyze data of electroencephalogram (EEG) collected during sleep to provide insightful information about an individual's sleep status such as polysomnography scoring. EEG signals are also used by human experts to create a hypnogram (a progression of different sleep stages through time), using the reference methodology published by the American Association of Sleep Medicine (AASM). The common sleep stages classes are Wake, Rem, N1, N2 and N3.

As the manual scoring is time-consuming and complex, various machine learning models have been proposed. Inspired by the frequency domain processing of EEG signals, there are various Convolutional Neural Nets (CNNs) architectures which have been proposed. Tsinalis et al. [7] apply CNNs on EEG and electrocardiography (ECG) signals, achieving an F1 score of 81% and overall accuracy of 74% over all subjects.

There is a strong temporal correlation in sleep stages, and methods [8] have used the recurrent neural architecture to utilize the correlation for predictions. SLEEPNET [8] is a deep recurrent neural network (RNN) that uses expert-defined features to represent each 30-sec interval and learns to annotate EEG. It is trained on the largest physiology database assembled to date, consisting of PSGs from over 10,000 patients from the Massachusetts General Hospital (MGH) Sleep Laboratory. The optimal model had 5 layers of LSTM cells with tanh activation function and dropout keep probability of 0.9. They achieved an average accuracy of 86%, which is comparable to human-level scoring performance.

Another approach is to use recurrent-convolutional neural networks (RCNN), where the CNN extracts sleep-specific subject-invariant features from RF signals and the RNN captures the temporal progression of sleep [9], [19], [20]. In [9], an adversarial training regime is adopted that discards extraneous individual or measurement condition specific information. This approach is done on a radio signals dataset and achieves an accuracy of 79.8%.

To capture higher order data from the input dataset, transfer learning was applied in [10]. Spectrograms are generated from electroencephalography, electrooculography, and electromyography. In the optimal model, the output from the spectrograms is fed as the input to the convolutional neural layers and then recurrent neural network (LSTM) layers. This achieved a weighted F1 score of 87%.

Another approach is to combine deep learning models with expert defined rules using a prototype learning framework to generate simple interpretable models [11]. This consists of the signal embedding module, where a CNN is used to get feature representation from raw polysomnogram (PSG) data. Next, there is an expert rule module that encodes each epoch into a multi-hot vector. Finally, there is the prototype learning module, where the output from the previous two modules are combined [11]. Ensemble learning with stacked sparse autoencoders can also be used to classify sleep stages. In [12], class-balanced random sampling is also used to achieve a F1 score of 84%.

We extend the RCNN approach [8], [19], [20] by using CNN to extract features and Transformers to leverage neighboring temporal context to predict sleep stage labels.

Approach/Metrics

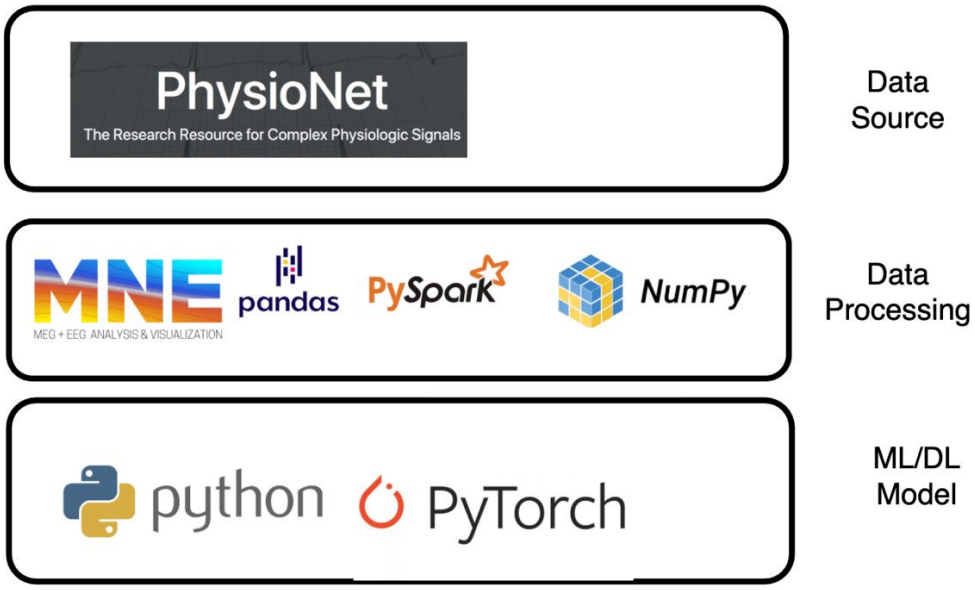

Figure 1 Technology Stack

Data

In this project, we used the Sleep-EDF Database Expanded from PhysioNet, which is an openly accessible database containing 197 whole-night Polysomnographic (PSG) sleep recordings with various channels. Along with the PSG raw signals, there are corresponding hypnograms labeled by human experts manually. Following from previous works, in this project, we focus on building machine learning and deep learning models with better accuracy in sleep stage scoring with respect to the human-labeled ground truths.

There are 197 whole-night recordings, each of which lasts around 10 hours, and there are 100 recordings of signals per second. Therefore, in total, there are approximately 709,200,000 data entries in this project.

ETL Pipeline

We collected Polysomnography (PSG) data from 197 study subjects over the whole night from PhysioNet 2018 [13] and select three one-dimensional channels in the electroencephalogram from each object: EEG Fpz-Cz channel, EEG Pz-Oz channel, and the EOG horizontal channel. We also obtained the corresponding labeled hypnogram for each study subject as the ground truth labeling. The raw channel signals are recorded as microvolts (μV) for each recording, and there are 100 recordings per second for a given subject. The corresponding hypnogram labels are given over a period of time as a tuple of starting time and duration.

We first extracted the signals in these three channels from the PSG file using the MNE library (a package that extracts the neurophysiological data such as EEG). We created indices to represent time for each discretized signal for every channel. Simultaneously, we extracted the corresponding sleep stage labeling from the hypnogram and for each discretized time, we assigned the corresponding label.

Since there are over 23GB of data, we decide to use Spark to manipulate and transform our data into appropriate formats for model training and testing. We created two types of RDDs in Spark for all data collected: raw signal RDD and label RDD. The raw signal RDD contains the patient ID together with the recorded signal strength in microvolts for discretized time, and each channel has its own raw signal RDD. The label RDD contains the patient ID, time, sleep stage label, and the corresponding start time for the label.

Table 1 Raw signal RDD Example

Time	PatientID	Signal
...	...	...
200	SC4001E0	0.89576247
...	...	...
300	SC4092E0	0.13987650

Table 2 Label RDD Example

Time	PatientID	SleepStage	StartTime
...	...	...	...
200	SC4001E0	W	120
...	...	...	...
300	SC4092E0	REM	240

Since we are only interested in labeling five sleep stages: Wake (W), Rem, N1, N2 and N3, we remove the other stages (e.g. “stage ?”, “stage walking time”, etc.) from the label RDD as well as the corresponding entries in the raw RDD. In addition, we divided the channel data into 30-second epochs (each epoch corresponds to one single sleep stage label) as our goal is to predict the epoch’s sleep stage labeling using the signals from each epoch. Lastly, we relabeled the sleep stages into numerical vectors, and we output the channel signal files into NPZ format for the training and testing of models.

Model Architecture

We use a convolutional neural networks (CNNs) (Figure 1) to extract time invariant features. It then uses a sequence to sequence architecture with Transformer models to learn long and short-term context of sleep stage labels from recent sleep epochs in order to more accurately predict sleep stages at each 30-second sleep epoch.

The CNN component of the model consists of 2 sections. One has a small filter to capture recent data i.e. data from the past several sleep epochs, while the other has a large filter data to capture long-term frequency trends of various sleep class labels. Both sections comprise of 4 1-dimentional convolutional layer. The first and last of these layers are followed by MaxPool layers. The details on filter sizes, number of output channels and stride size is given in Figure 1. The two CNN sections, represented as (a) and (b) respectively in Figure 1, differentiate in their filter sizes and their outputs are concatenated along axis of the second dimension to create the extracted feature.

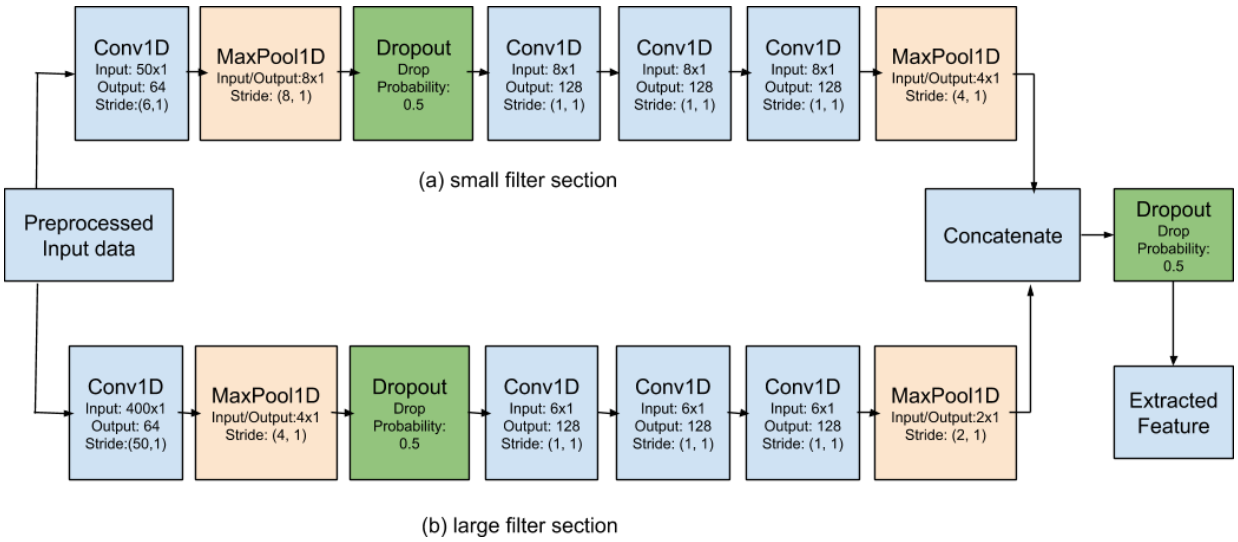

Figure 1. CNN model architecture. Each 1-dimentional Convolutional layer is followed by a ReLU layer.

The extracted features from the CNN model are then passed into a Transformer model to predict the sleep stage label for any given 30-second time epoch after calculating the probability for the likelihood of a given sleep class to follow a sequence of sleep classes. A sequence of classes is passed in with the positional encoding layer to incorporate data regarding the order of the sleep classes. This model relies on attention mechanism to draw global dependencies between EEG input and one of 5 sleep stage output predictions.

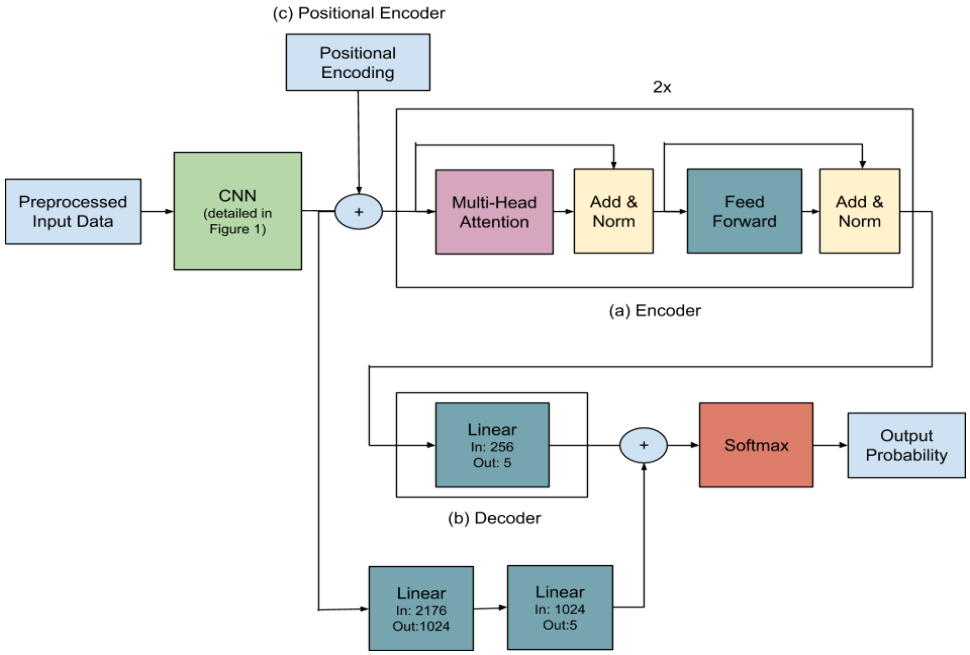

Figure 2. Feature Extraction and Transformer Sleep Class Predictor Model with Residual Connections. The model specifications for (a) the encoder and (c) the positional encoder is outlined in [17] in detail. The "2x" above the encoder indicates that there are two encoder layers.

The Transformer model consists of 3 main components in its architecture: the encoder, the decoder and the positional encoder.

- The encoder is composed of a stack of 2 identical layers. Each Transformer Encoder Layer is created with 2176 expected features in the input, 2 heads in the multi-head attention models, 128 is the dimension of the feedforward network, and a drop out probability of 20 percent is used.
- The decoder is similarly composed of a stack of 2 identical layers. Each Transformer Decoder Layer consists of a simple linear layer where the size of each input sample is 2176 and the size of each output sample is set to the number of sleep stages i.e. the labels that are being predicted. In our case, there are 5 sleep stage cycles. Currently, we are using a mask to restrict our model to only use sleep cycle labels from earlier positions in the sequence to influence the prediction of the sleep stage at our current time step. Any tokens on future positions are masked.
- The positional encoding is used to inject information about the local or global position of the sleep cycle class token in the sequence of class tokens. We implemented the positional encoding introduced in the original Transformers paper [17]. See reference for implementation details on this. Generally, this function is using sine and cosine functions of different frequencies and is useful because positional encoding values for any fixed offset k is found to be a linear representation of the positional encoding at the current time step.

Experimental Results

Instead of training the whole network end-to-end, we first train the CNN only model and achieve an accuracy of 67% and mean F1 score of 71%. We then load these pretrained CNN weights into the CNN+Transformer and train the model for 20 epochs using stochastic gradient descent optimizer with gradient clipping. We observed that the input data had a class imbalance among the 5 sleep stage classes. We implemented a weighted cross-entropy loss, where weightage was assigned as the inverse of class frequency. The accuracy results for various models is noted in Table 3. Additionally, the loss and accuracy curves for our optimized model is shown in Figure 3.

Table 3. Accuracy Results for various models: the baselines Stacked Sparse Autoencoders [21] and SleepEEGNet [19], our CNN-only component, and our final CNN+Transformer model

Method	**Accuracy**	**Overall F1**	**Mean-F1**	**W-F1**	**N1-F1**	**N2-F1**	**N3-F1**	**REM-F1**
Stacked Sparse Autoencoders [21]	78	-	**84**	81	**60**	78	**89**	**80**
SleepEEGNet [19]	**80.03**	-	73.55	**91.72**	44.05	**82.49**	73.45	76.06
CNN only	67	71	67.26	88.10	44.41	70.21	66.61	67.00
CNN-Transformers	75	78	75.05	90.44	52.16	78.36	76.56	77.57

We use a held-out test set to compute metrics like accuracy, and f1-score (overall and per-class) (Table 1). Although we have a lower accuracy compared to the baseline, we achieve a higher F1 score on less-frequent classes and as a result, we achieve a higher mean F1-score.

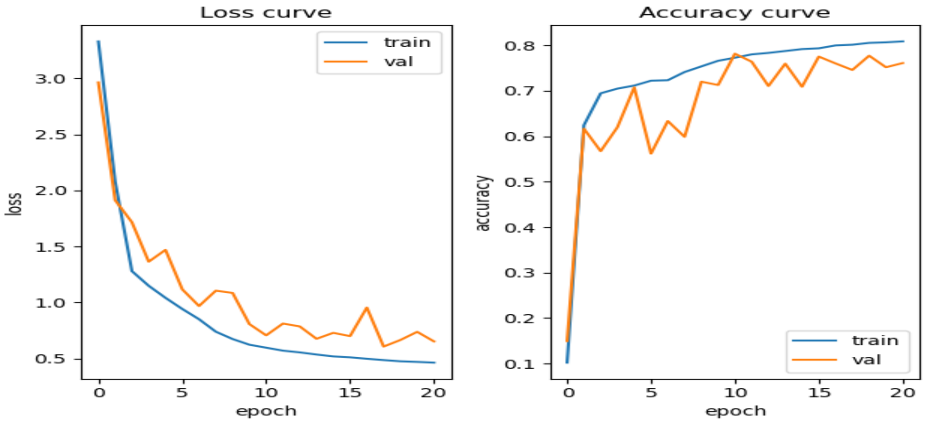

Figure 3. Loss Curve (left) and Accuracy Curve (right) on the training and validation set for 20 epochs of training

Discussion

Our model was able to get reasonably high accuracy but has not surpassed the baseline accuracy yet. We believe that we were able to reach relatively high accuracy so far for several reasons. Our feature extraction approach performs well because we leverage CNNs to extract time invariant features. Furthermore, instead of using just one CNN model, we learn a CNN model with two sections. One has a small filter to capture recent data i.e. data from the past several sleep epochs, while the other has a large filter data to capture long-term frequency trends of various sleep class labels. For feature extraction, concatenating the output from both these models instead of just one allows our models to learn long and short-term context of sleep stage labels from recent sleep epochs in order to more accurately predict sleep stages at each 30-second sleep epoch.

Another reason for our fair performance is because we used Transformers model architecture to predict the sleep cycle class at each time step i.e. 30-second intervals. Alternatively, we could have used RNNs [15] which has widely been explored in automated sleep scoring in the past since RNNs have the ability to preserve useful information across time and hence are quite adept at handling temporal sequences. Unfortunately, these vanilla RNNs are unable to distinguish what is useful information from past learning, instead treating each past sequence data with equal importance. Long-Short-Term-Memory [16] are a special type of RNNs and are better suited than vanilla RNNs because they remember which past data is important and which is worth forgetting. However, in the NLP domain, transformers and attention [17] have overtaken LSTMs and established themselves as the new cutting-edge model architecture. Thus, our model for this sleep stage class prediction task is leveraging the most state of the art model from a related, NLP domain.

The reason why Transformer models are best suited for this task is because of Transformer's Attention mechanism, which given an input sequence, decides at each time step which other parts of the sequence are important. Due to its encoder-decoder structure, Transformers' Encoders retain keywords i.e. subsequences which are important and communicate this to the Decoders. Furthermore, Transformers takes multiple possible input attention weightage into account at the same time. Thus, Decoders are able to able to compute the sleep cycle class after processing the input encoded by the encoder and the attention mechanism. Transformers are also quicker than equivalent LSTMs and other RNN architectures.

We removed the mask in our Transformer Encoder that was being used to restrict our model to only use sleep cycle labels from earlier positions in the sequence to predict the sleep stage at our current time step. When we want to predict the sleep cycle label at a particular time step, we take a look at both the data before and after this particular timestamp. This is helpful in avoiding predictions which are unrealistic. For example: a timestamp of label 'awake' between two labels of 'REM' is not possible in the real world. A bidirectional view of the data should help us in these cases. Despite making intuitive sense, this approach did not empirically improve our accuracy and F1 scores significantly.

Conclusion

Challenges: Throughout the execution of this project, we faced various challenges. Firstly, we initially underestimated the size of our data; therefore, we realized that it is hard to process all data at once. Nevertheless, we have to optimize the disk usage and evaluate the trade-offs between using disk space (i.e. output raw signals into CSV format then have PySpark read in these CSV files) versus using RAM (eliminate the output of CSV step and directly process data with PySpark). In the end, we changed the architecture of the ETL pipeline and have each study subject output their own signals and labels so that we don't have to process all data at once. With this approach, we are able to process the data in batches and obtain a balance between using disk space (need more disk space, also faster) versus using RAM (need more RAM, also slower). Secondly, writing customized functions in Spark can be tedious while these customizations can be done easily with NumPy or pandas. Therefore, we decided to use NumPy to customize the indices and let Spark to handle the rest of the actions (i.e. joining RDDs, grouping, eliminating values, etc.). We faced challenges with the model architecture. When we tried to increase model complexity to improve performance, our model would overfit.

In conclusion, our model is performing moderately well right now but was unable to surpass the baseline models discussed. Our dataset has a significant class imbalance problem. When there are significantly more instances of a one class versus another, the model learns to skew prediction results in favor of the high frequency class in an attempt to increase its accuracy. Thereby not assessing the data to make a fair call on the class prediction. In order to counter this, for future works, we recommend applying focal loss [18] with class balancing factor on our network. Focal loss was introduced in an object-detection paper and it performs well even when there is large class imbalance. Since our class imbalance is not as pronounced as the object detection scenario,

applying a class balancing factor will be appropriate. Furthermore, since our dataset was small, so it may be fruitful to train on a very large dataset where we can adjust the class imbalance by generating equal number of samples.

We attempted to improve our accuracy through various ways. We tried switching to spectral domain, adjustments to model architecture, in addition to tuning the hyper-parameters. However, we were unable to significantly improve the model accuracy. This is why we would recommend trying ensemble learning approaches in the future. Ensemble approaches have historically outperformed results from any single model architecture in the ensemble. Furthermore, [21] has performed significantly well with ensemble learning as well.

Contributions

Ayush Baid: Read papers, designed model architecture, implemented the model, ran model to get accuracy results.

Ria Verma: Read papers, designed model architecture, wrote details of survey results, model architecture, results, discussion etc. in project proposal/draft/report.

Yujia Xie: ETL pipeline and data processing with Spark.

Individual Report

Participation among group members: all teammates participated roughly equally.

References

1. Newman AB. Relation of Sleep-disordered Breathing to Cardiovascular Disease Risk Factors: The Sleep Heart Health Study. *American Journal of Epidemiology*. 154(1): 50–9, 2001.
2. Punjabi NM. Sleep-Disordered Breathing, Glucose Intolerance, and Insulin Resistance: The Sleep Heart Health Study. *American Journal of Epidemiology*. 160(6): 521–30, 2004.
3. Ju ES, Lucey BP, Holtzman DM. Sleep and Alzheimer disease pathology—a bidirectional relationship. *Nature Reviews Neurology*. 10(2): 115–9, 2013.
4. Liu Y, Croft JB, Wheaton AG, et al. Clustering of Five Health-Related Behaviors for Chronic Disease Prevention Among Adults, United States, 2013. *Preventing Chronic Disease*. 2016;13.
5. Acharya UR, Faust O, Kannathal N, Chua T, Laxminarayan S. Non-linear analysis of EEG signals at various sleep stages. *Computer Methods and Programs in Biomedicine*. 80(1): 37–45, 2005.
6. Fraiwan L, Lweesy K, Khasawneh N, Wenz H, Dickhaus H. Automated sleep stage identification system based on time–frequency analysis of a single EEG channel and random forest classifier. *Computer Methods and Programs in Biomedicine*. 108(1): 10–9, 2012.
7. Tsinalis O, Matthews PM, Guo Y, and Zafeiriou S. Automatic sleep stage scoring with single-channel eeg using convolutional neural networks. *arXiv preprint arXiv:1610.01683*, 2016.
8. Biswal S, Kulas J, Sun H, Goparaju B, Westover BM, Bianchi MT, and Sun J. SLEEPNET: Automated sleep staging system via deep learning. 26 July 2017.
9. Zhao M, Yue S, Katabi D, Jaakkola TS, Bianchi MT. Learning sleep stages from radio signals: A conditional adversarial architecture. *In International Conference on Machine Learning*, pages 4100–4109, 2017.
10. Zhang L, Fabbri D, Upender R, Kent D. Automated sleep stage scoring of the sleep heart health study using deep neural networks. *Sleep*. 42(2): 1—10, 2019.
11. Al-Hussaini I, Xiao C, Westover MB, and Sun J. Sleeper: interpretable sleep staging via prototypes from expert rules. *In Machine Learning for Healthcare Conference*, pages 721–739, 2019.
12. Tsinalis O, Matthews PM, and Guo Y. Automatic sleep stage scoring using time-frequency analysis and stacked sparse autoencoders. *Annals of Biomedical Engineering*. 44(5): 14 October. 2015.
13. PhysioNet: The Sleep-EDF database [Expanded]: http://www.physionet.org/physiobank/ database/sleep-edfx/ (Accessed January 2015).
14. Iber C, Ancoli-Israel S, Chesson A, and Quan SF. The AASM Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and Technical Specifications. Westchester, IL: *American Academy of Sleep Medicine*, 2007.
15. Mikolov T, Karafiat M, Lukas B, Cernocky J, Khudanpur S. Recurrent neural network-based language model. *Eleventh annual conference of the international speech communication association*, pages 1045—1048, 2010.
16. Hochreiter S, and Schmidhuber J. Long short-term memory. *Neural computation.* 9(8): 1735-1780, 1997
17. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. *Advances in neural information processing systems*. Pages 6000-6010, 2017.
18. Lin T.-Y., Goyal P., Girshick R., He K., & Dollar P. Focal Loss for Dense Object Detection. IEEE International Conference on Computer Vision (ICCV), 2017.
19. Mousavi S., Fatemeh A., and Acharya U. R. "SleepEEGNet: Automated sleep stage scoring with sequence to sequence deep learning approach." PloS one 14.5, 2019.
20. Supratak, A., Dong, H., Wu, C., & Guo, Y. DeepSleepNet: a model for automatic sleep stage scoring based on raw single-channel EEG. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 25(11), 2017.
21. Tsinalis, O., Matthews, P. M., & Guo, Y. Automatic Sleep Stage Scoring Using Time-Frequency Analysis and Stacked Sparse Autoencoders. Annals of Biomedical Engineering, 44(5), 1587–1597. 2015.