ElderNet: Automated Electroencephalography Sleep Stage Scoring for Elderly Patient Populations

Michael Silvernagel, Spiros Baltsavias, Abhijeet Phatak
[mpsilver, sbaltsav, aphatak] @stanford.edu

Abstract

Sleep is an integral component of healthy living. Poor sleep quality is associated with obesity, diabetes, high blood pressure, stroke, and cardiovascular disease, while sleep deprivation has been shown to increase one's susceptibility to cancer, depression, and memory loss [1, 2]. Despite this, sleep-related problems remain an under-recognized public health issue, with an estimated 50 to 70 million Americans suffering from sleep disorders [2]. Those over the age of 60 compose a significant portion of this group, with 40% to 50% of older adults reporting disturbed sleep, and 28% exhibiting chronic insomnia [3]. Given age-related changes in sleep architecture and the associated difficulties they present for sleep stage scoring, our project examines how deep learning can be used to facilitate data analysis for simplified polysomnography setups, allowing older adults to easily and accurately access this valuable health information.

1 Introduction and Related Work

The clinical standard for characterizing sleep disorders is nocturnal polysomnography (PSG), during which a patient's sleep is monitored and scored into stages. Several inadequacies, however, currently exist with PSG: **(1)** Patients are attached to upwards of 11 types of wired sensors, creating an uncomfortable and unnatural sleeping environment (Fig. 1); **(2)** Complex PSG systems require setup by experienced technicians, greatly increasing costs; **(3)** PSG recordings are scored by human technicians, making the analysis time-consuming and highly subjective.

To simplify PSG setups and remove the need for human technicians and scorers, numerous automated sleep scoring methodologies operating on a subset of PSG sensor data—primarily electroencephalography (EEG) data—have been proposed [4]. More recently, researchers have utilized deep neural networks trained on raw EEG data to create robust classification schemes [5, 6]. While these neural network classification approaches are promising, to our knowledge, their efficacy on older patient populations has not been studied. Sleep architecture undergoes significant changes with age. Older adults experience more fragmented sleep and an increased incidence of arousals; as a result, time spent in deeper sleep stages (e.g. N3, REM) decreases [3]. Additionally, with age, two important markers of sleep stage activity change: slow-wave EEG activity decreases, and sleep spindles—high-frequency bursts of EEG activity—decline in amplitude, incidence, frequency, and duration [7]. The class imbalance and reduced signal-to-noise ratio produced by age-related changes in sleep architecture pose additional challenges when creating automated sleep-scoring algorithms. Using deep learning frameworks, our project looks to achieve sleep stage classification accuracy for older adults that approaches human level scoring accuracy.

Figure 1: PSG Setup

2 Dataset

Our project relies on PSG data provided and scored by Dr. Makoto Kawai, a Stanford sleep clinician and assistant professor in psychiatry. With 85 patients (67.2 $\pm$ 5.7 years), recording times of approximately 8 hours, and sleep staging scored on 30 second intervals, we have $\sim$130,000 epochs of labeled data. In line with recent efforts aimed at simplifying PSG, we limited our analysis to 4 channels of EEG: C3-A2; C4-A1; O1-A2; O2-A1. Depicted in Fig. 2, these channels were placed in accordance with the International 10-20 system.

The numerical labels for each epoch, which were manually generated by Dr. Kawai, were as follows: 0 = awake; 1 = N1; 2 = N2; 3 = N3; 4 = REM; 5 = unscorable (too noisy to score, likely due to subject movement or non-optimal electrode attachment). As is commonly done in literature, our first step processing this dataset was to remove the examples that were labeled unscorable, which resulted in a truncated dataset size of 83,193 examples. Given this moderately large dataset we decided to do an approximately 80-10-10 train-dev-test split by placing 68 patients in the training set, 9 patients in validation set, and 8 in test test. We did not distribute examples from an individual patient to different datasets, thus ensuring that our algorithm sees new patients in test time.

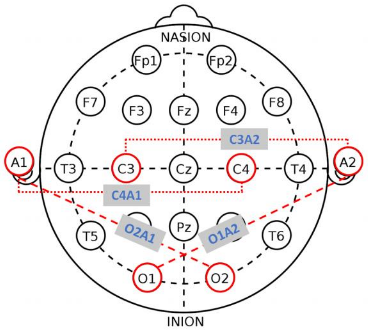

Figure 2: EEG Channels

To better understand our data, we plotted histograms of the data of our train, dev, and test sets (Fig. 3). We noticed that the majority of our labels were representative of the N2 sleep, and that labels for the N3 stage were very few. We recognized this uneven distribution of labels might result in poor classifier performance, motivating the use of weighted cost functions as described later.

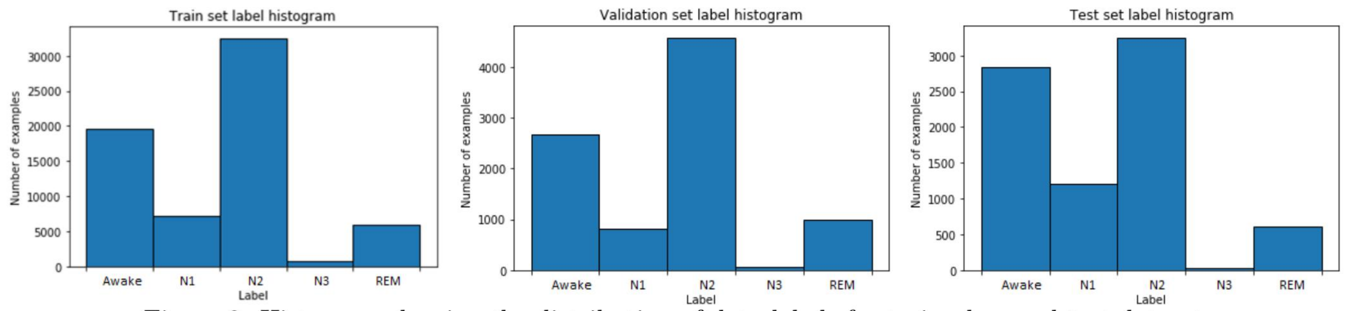

Figure 3: Histogram showing the distribution of data labels for train, dev, and test datasets

3 Input pipeline

Our dataset consists of .edf files containing approximately 8 hours of PSG sensor data for each subject, along with .xml files containing sleep stage labels. MATLAB scripts were created to extract the EEG data from the .edf file, divide the EEG data into 30-second epochs to align with the human scoring procedure, and match each segment with its corresponding label. As the data was sampled at 256 Hz, each 30-second epoch contains 7680 time points, and 4 channels (data recorded from 4 pairs of electrodes) were available. An example 30-second waveform from 4 channels is shown in Fig. 4. After reshaping our data, our input 'X' dimensions were m x 7680 x 4, where m is the number of examples (30-second segments) and our 'Y' label dimensions were m x 5, where the 5 dimensions correspond to a one-hot encoding of the numerical labels.

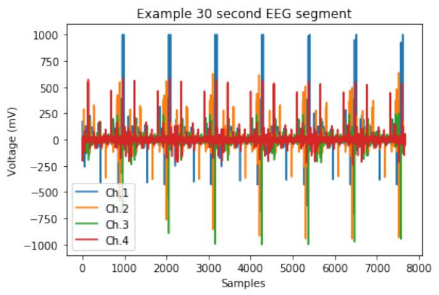

Figure 4: Sample EEG waveform

Moving from our milestone subset of 5 patients to the full 85 patient dataset, we found loading all of the examples in memory before running our training resulted was not possible. We instead implemented a data generator to form batches of a desired size on the fly during model training/evaluation. This allowed us to use the full dataset and utilize the available GPUs efficiently.

4 Preprocessing

Modern neural network architectures such as in [5, 6] consist of multiple convolutional neural networks at the input to perform feature extraction, followed by recurrent neural networks to learn sleep classification schemes. In this project we decided to investigate whether feature extraction could be done as a preprocessing step to simplify our network architecture and speed up our models. We investigated two preprocessing methods: wavelet coefficient computation and statistical analysis, and spectrograms.

4.1 Wavelet coefficients

The wavelet transform is a powerful method for time-frequency analysis generated by convolving a signal with a localized, wave-like oscillation, or wavelet. By scaling and time shifting the wavelet, frequency and time characteristics of the signal can be accurately captured. Additionally, since the wavelet is localized in time, it can accurately detect sudden signal changes, providing a distinct advantage over the Fourier transform. Since EEG sleep scoring is highly dependent rapidly varying time and frequency signal characteristics, the wavelet transform is a natural tool for EEG signal analysis. For this analysis, an 8 level wavelet decomposition was performed using the second order Daubechies wavelet. To capture the signal characteristics, simple statistical analysis was performed on the 3rd through 8th level detail coefficients of the wavelet decomposition. As the original EEG signal was sampled at 256 Hz, these detail coefficients correspond to frequency bands which align with EEG frequency bands of interest for sleep. On these detail coefficients, a variety of statistical measures were performed, and through heuristic testing, the combination of root mean square, variance, skew, and kurtosis calculated over the detail coefficients generated the best classification.

4.2 Spectrograms

A spectrogram is another way of representing the frequency content of a signal as it changes in time—in this case, the signal is windowed at different time segments and Fourier transforms are performed over each window. The result is an image representing the frequencies (y-axis) present in the signal and their amplitude (color), vs. time (x-axis). The idea behind the use of spectrograms was inspired by the high performance of convolutional neural networks in recent literature for computer vision and other applications involving image data. Here we applied spectrograms to the EEG channel data to essentially produce 2D time-frequency maps similarly to [8], with the equivalent of RGB channels provided by the 4 EEG channels. Our hypothesis was that spectrograms in combination with a CNN architecture could lead to efficient learning of relevant time-frequency features that could help classify EEG sleep stages. We also investigated whether RNNs in combination with CNNs can also be used to further extract features along the time axis. To compute the spectrograms we used the SciPy function spectrogram with an input sampling frequency parameter 256 Hz and a default Tukey window with shape parameter of 0.25. Since the range of frequencies we considered relevant were 0.5 to 32 Hz, we cropped the spectrogram frequency content from 128 Hz down to 32 Hz. An example output image for a 30-second epoch is shown below in Fig. 5.

5 Model implementation

We decided to implement our training and evaluation platform on Keras (with Tensorflow backend) for rapid prototyping.

In all models, the output layer was a softmax layer with 5 classes. Since this is a multi-class classification problem, we defined our cost function as the categorical cross-entropy loss. We also decided to use class weights to account for class imbalance with our data. In this case, for weight w corresponding to each class out of C, the cost function is defined as:

$$J(y_{true}, y_{predict}) = \sum_{i}^{m} \sum_{k}^{C} -w_k y_{true}^{(k)} \log{(y_{predict}^{(k)})} \tag{1}$$

Given our choice of preprocessing steps we decided to divide our efforts into 4 models: Baseline + FC; Wavelets + RNN; Spectrograms + CNN + RNN; and Spectrograms + CNN + Attention.

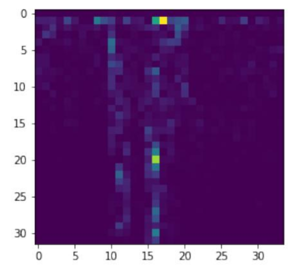

Figure 5: Example spectrogram image. X-axis - time, Y-axis - frequency.

5.1 Baseline model

We decided to try a simple baseline model consisting of a fully connected network. In this model each input example with dimensions 7680 x 4 was flattened and put through layers with ReLU activation. The first 6 layers were kept fixed with dimensions 2048, 1024, 512, 256, 128, and 64 respectively. As a hyperparameter in our milestone we had varied the number of 64-size layers following the first 6 layers: 4, and 24 were used for initial tests. Here we fixed this parameter to 24 and did not perform any further tuning. This model was used to get an intuitive sense of the performance of an end-to-end simple network, i.e. without using any manual feature extraction steps.

To train our network we used an Adam optimizer with default parameter values of $\alpha(0.001), \beta_1(0.9), \beta_2(0.999)$, and $\epsilon(1e^{-7})$. We used a mini-batch size of 128 examples and trained for 100 epochs.

5.2 Wavelet coefficients and RNNs model

With the root mean square, variance, skew, and kurtosis calculated on six levels of wavelet detail coefficients, the input dimensionality was reduced to m x 24 x 4. This input was fed into a variety of deep learning architectures, including fully connected, convolutional, and recurrent layers. The use of recurrent layers provided the best performance, and treatment of the wavelet features as time steps allowed the model to learn relationships between EEG frequency bands for varying stages of sleep. As LSTMs provided a slight improvement over RNN and GRU layers, the final architecture consisted of feeding the input into a bidirectional LSTM with 512 units, followed by a dropout layer with a dropout rate of 0.5 to reduce overfitting. The full LSTM sequence was returned and input into another 512 unit bidirectional LSTM and dropout layer. Again, the full LSTM sequence was returned, and subsequently flattened and fed into a five unit fully connected layer. A softmax activation was then applied to determine the output classification. Architectures with additional stacking of bidirectional LSTM layers—with and without skip connections—before the fully connected layer were explored, but they did not provide any performance advantage.

5.3 Spectrogram and CNN/RNNs model

In this model the input raw EEG waveforms are converted into spectrograms (one per channel), with final dimensions m x 32 x 34 x 4. Two 2D convolutional layers were implemented at the model input to extract features from the spectrum images and reduce dimensionality. For the next section of the model different RNNs were implemented and evaluated: LSTMs, GRUs and stacked versions of both. The addition of stacked RNNs did not improve performance, and so we decided to proceed with the single LSTM layer. Finally a two layer fully connected network (ReLu activations) was added before the output layer, with dropout after each layer to reduce variance.

5.4 Spectrogram + CNN + Multi-Headed Attention model

Attention is a nice way of limiting the number of trainable parameters and yet learn very complex representations. Attention is widely used in Neural Machine Translation (NMT) and other sequence-to-sequence (seq2seq) problems. To improve our training accuracy, it is important that we either use deeper network, but then we need more data to avoid over-fitting. Specifically, we implemented the multi-head attention model as described in [9] after two convolutional layers and then had an LSTM to model recurrent behavior in the attention maps connected ultimately to a FCN. We also tried other architectures like few Residual-Inception blocks to see if gradient propagation is the reason why our models and saturating. However, we will limit our results and discussions to the aforementioned models.

6 Results

6.1 Metrics

Here we used categorical accuracy, which is defined as the number of matches between predicted and actual labels in the evaluation test set, divided by the total number of evaluation examples. Our accuracy target was originally defined as 0.8, since that is the typical human accuracy standard in clinical settings.

6.2 Baseline model

Similarly to our milestone results, this model resulted in accuracy values which capped at 0.5 for the train and validation set. The maximum accuracy is reached within the first couple of epochs. Our test accuracy was 0.41 - we checked the output predictions of our model and saw that they consist only of labels of a single category, the one with the largest number of examples. This implies our model is not able to learn a better prediction strategy other than choosing the category with the largest number of examples. A confusion matrix on our test set predictions and true labels confirmed this result (not shown here). The results presented here used the weighted loss function, which was not found to improve results over the conventional cross-entropy loss function.

6.3 Wavelet model

This network utilized the Adam optimizer with default parameter values of $\beta_1(0.9), \beta_2(0.999)$, and $\epsilon(1e^{-7})$, as well as the weighted cross entropy loss, where weights were adjusted based on the relative class distributions. A hyperparameter search was performed to determine the optimal learning rate and mini-batch size; this produced a learning rate of $\alpha_1(0.0005)$ and mini-batch size of 1024. The final model was then trained for 50 epochs.

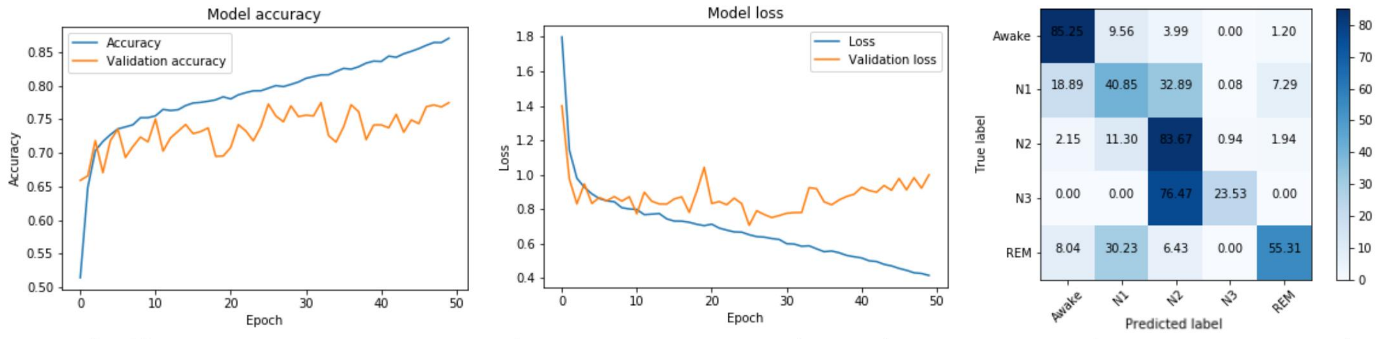

Figure 6: (Left) Example accuracy vs. epoch for our wavelet model. (Middle) Loss vs. epoch for the same model. (Right) Normalized confusion matrix - test accuracy was 0.75 for this model.

Results from this model are depicted in Fig. 6. Accuracy for the validation and test sets was about 75%. This level of accuracy was achieved after around 25 epochs of training, at which point the model begins to overtrain. Excellent accuracy was achieved for the detection of the majority classes, awake and N2, but the model suffered when predicting minority classes, particularly N3. One possible remedy for this issue would be to collect additional training data. It is also worth mentioning that because the wavelet representation of the data allowed for dramatic compression, it was much easier to develop and rapidly train large models, which might have contributed to the success of this particular approach.

6.4 Spectrogram model

To design our network we used an Adam optimizer with default parameter values of $\alpha(0.001), \beta_1(0.9), \beta_2(0.999)$, and $\epsilon(1e^{-7})$. We used a mini-batch size of 128 examples and trained for 100 epochs. For this report we then also performed learning rate hyperparameter tuning for our single LSTM based network (0.0001, 0.0005, 0.001, 0.01). For 0.01, we get similar results as the baseline model, where the accuracy saturates at 0.5. For lower learning rates, we get results as shown in Fig. 8 (0.001 used here). Despite the inclusion of dropout layers we note the model does not generalize to the validation dataset and starts to overfit on the train set. Our validation and test accurate saturate at around 0.72. The confusion matrix shows the algorithm achieves good accuracy for the majority classes (Awake and N2), but for example misses all N3 cases since the dataset does not contain many such examples, leading us to similar conclusions as for our wavelet model.

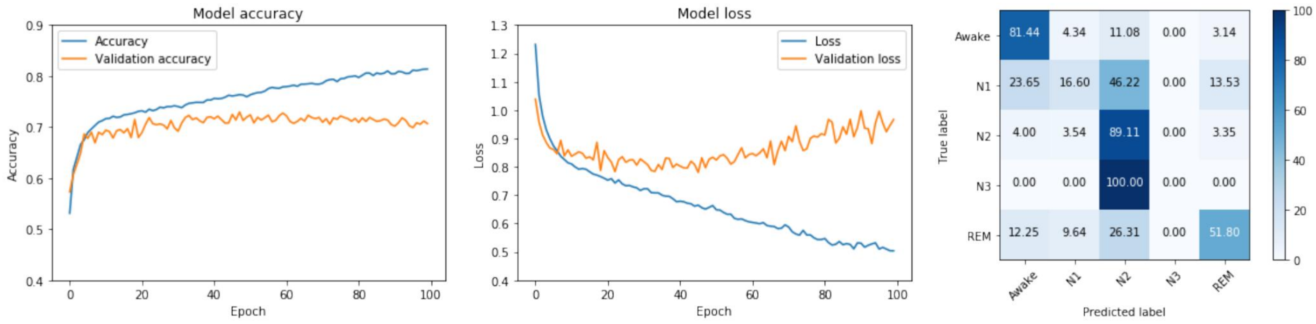

Figure 7: (Left) Example accuracy vs. epoch for our spectrogram model. (Middle) Loss vs. epoch for the same model. (Right) Normalized confusion matrix - test accuracy was 0.72 for this model.

6.5 Attention model

This model had only a few deviations from the earlier model in that, after the convolutions, we employ a multi-head self attention network as described in literature. The mini-batch size was 128 and was trained for 100 epochs with Adam optimizer with learning rate of $1e^{-3}$. The reason for trying out this model was that with fewer parameters, one can model seq2seq problems efficiently. We saw that it indeed performs well. Given more time, we would like to analyze the class wise attention maps which can make the model more interpretable as it can tell us which kind of patterns in the spectrogram are to be focussed on. We also tried Residual and Inception (ResNet) architectures but all of these seemed to saturate at a validation and test accuracy of about 0.72.

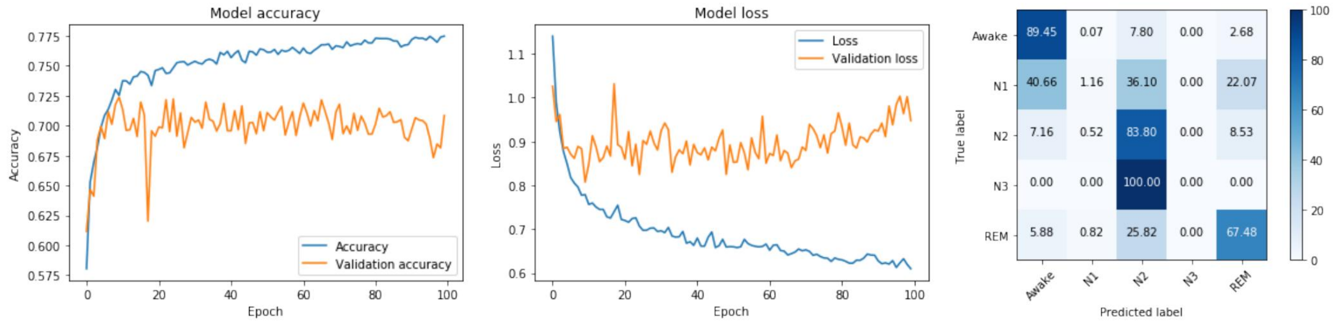

Figure 8: (Left) Example accuracy vs. epoch for our attention spectrogram model. (Middle) Loss vs. epoch for the same model. (Right) Normalized confusion matrix - test accuracy was 0.72 for this model.

7 Discussion and Conclusion

In this project we implemented several deep learning models to achieve automatic sleep stage classification for elderly patients. Since the project milestone we were able to design three networks achieving over 0.72 accuracy for a classification task that humans are expected to perform 0.8 or better. Using wavelet coefficients as a preprocessing method we were able to get our best test accuracy of 0.75 thanks to the dimensionality reduction and robust time-frequency map representation.

Based on our interpretation of the results, we believe a larger dataset could help us reduce our bias. The addition of regularization then could also bridge the gap between our train and dev/test performance. Given more time we would also like to perform finer architecture and hyperparameter searches within these models, and evaluate performance using more complete composite measures along with per class statistics (precision, recall, F1-score). A future direction can be to make these models more interpretable so that they can be used as an aid to sleep study researchers in annotating. Another exciting idea is to use knowledge distillation to train a student network (smaller) which will lead to compact deployable models especially for wearable devices.

8 Contributions and Acknowledgments

Spiros Baltsavias was responsible for implementing the spectrogram preprocessing step and spectrogram + CNN/RNN model. He also evaluated the baseline fully-connected model and worked on improving/debugging Abhi's data generator for efficient data loading during run time. Michael Silvernagel was responsible for development of MATLAB programs that extracted the EEG data from the .edf files, divided the EEG data into 30-second epochs, and matched each segment with its corresponding label. He also implemented the wavelet preprocessing and was responsible for the wavelet coefficients + RNN model. Abhijeet Phatak setup AWS with remote jupyter notebook `ssh`-ing, implemented the data generator network then improved by Spiros and Michael. He also worked on the multi-head self attention model. We would like to thank Dr. Makoto Kawai for providing the dataset and labels, and our project mentor Aarti Bagul for her helpful guidance and advice. We also thank Amazon AWS for providing the necessary computational resources for our project and the CS230 staff for the course materials.

References

[1] M.H. Kryger, T. Roth, and W.C. Dement. *Principles and Practice of Sleep Medicine E-Book*. Elsevier Health Sciences, 2017. ISBN: 9780323377522. URL: `https://books.google.com/books?id=AB-KCwAAQBAJ`.

[2] S Fulda and H Schulz. "Cognitive dysfunction in sleep disorders". In: *Sleep medicine reviews* 5.6 (2001), pp. 423–445.

[3] Sonia Ancoli-Israel Jana R. Cooke. "Handbook of Clinical Neurology". In: Elsevier, 2014. Chap. Normal and abnormal sleep in the elderly, pp. 653–665.

[4] Khald Aboalayon et al. "Sleep stage classification using EEG signal analysis: a comprehensive survey and new investigation". In: *Entropy* 18.9 (2016), p. 272.

[5] Akara Supratak et al. "DeepSleepNet: A model for automatic sleep stage scoring based on raw single-channel EEG". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25.11 (2017), pp. 1998–2008.

[6] Alexander Malafeev. "Automatic Human Sleep Stage Scoring Using Deep Neural Networks". In: *Frontiers in Neuroscience* 12.781 (2018). DOI: `doi:10.3389/fnins.2018.00781`.

[7] Mirjam Munch. "EEG sleep spectra in older adults across all circadian phases during NREM sleep". In: *Sleep* 33.3 (2010), pp. 389–401. DOI: `doi:10.1093/sleep/33.3.389`.

[8] Giulio Ruffini et al. "Deep learning with EEG spectrograms in rapid eye movement behavior disorder". In: *bioRxiv* (2018). DOI: `10.1101/240267`. eprint: `https://www.biorxiv.org/content/early/2018/05/18/240267.full.pdf`. URL: `https://www.biorxiv.org/content/early/2018/05/18/240267`.

[9] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

Code libraries used for this project included scikit-learn, Tensorflow, and Keras.