Automatic Sleep Stage Classification

Illinois Media Space Link: https://mediaspace.illinois.edu/media/t/1_9ufzloeb

Pradeep Dwivedi
University of Illinois Urbana-Champaign
901 West Illinois Street, Urbana, IL 61801
pkd3@illinois.edu

Michael Renteria
University of Illinois Urbana-Champaign
901 West Illinois Street, Urbana, IL 61801
mrente5@illinois.edu

Jason Kolter
University of Illinois Urbana-Champaign
901 West Illinois Street, Urbana, IL 61801
jkolter2@illinois.edu

Zichun Xu
University of Illinois Urbana-Champaign
901 West Illinois Street, Urbana, IL 61801
zichunx2@illinois.edu

ABSTRACT

Sleep stage classification is important for sleep disorder diagnosis. Emerging deep learning studies in sleep stage classification have demonstrated high performance and can greatly reduce manual effort from human experts. In this paper we present and evaluate several machine learning models that automatically classify sleep stages. They include convolutional neural network (CNN) model, CNN and recurrent neural network (RNN) model, and selected non-deep learning models. All models are developed without any signal preprocessing or hand-engineered features. The CNN-RNN model achieved the best performance among all models with an accuracy of 0.79, average F1 score of 0.72, and cohen's kappa coefficient of 0.72. The labels produced by our model are in substantial agreement with that from a human expert.

1. INTRODUCTION

1.1 Problem Statement

Sleep disorders are widespread and can have a large impact on a person's health and quality of life [1]. Sleep stages are critical for diagnosing sleep disorders [2]. There are four stages of sleep that various studies analyze in order to find non regular sleeping patterns, and furthermore pinpoint various sleep related disorders. According to American Academy of Sleep Medicine (AASM), the four stages [3] are broken down into two categories: rapid eye movement (REM) sleep and non-REM sleep. Non-REM sleep is composed of stages N1, N2, and N3. Previously N3 was further divided into N3 and N4. REM sleep is a stage and category by itself. A polysomnogram (PSG) is a comprehensive sleep test that records various sensor readings of the body during a subject's sleep.

PSG recordings are divided into 30-second intervals called epochs, which serve as a basic time period for data analysis. A human expert then assigns a sleep stage to each of the epochs. Since, it typically takes 90 minutes to hit the fourth stage of sleep in a normal patient [4], it is needless to say that a human annotator often works with many hours of PSG data on each patient. The hours of data is then multiplied by each of the various sensors you choose to use to help predict the specific stages of sleep. This is time and labor intensive because sleeping stages are scored based on a collection of rules given by AASM [3, 5]. They include (but are not limited to) signal intensity, patterns, duration, sequence or co-occurrence of signals and so on.

1.2 Related work

Deep learning methods have shown great promise in generating sleep stage labels [6-8] and some models have achieved comparable accuracy to human experts. It can greatly aid the process of diagnosing sleep disorder.

Convolutional neural network, usually seen in image recognition, has also shown great success in signal processing like EEG classification, arrhythmia detection and motor-fault detection [9]. It has many advantages. For example, it can be applied to raw signal data without pre-processing. It can learn to extract useful spatial and time-invariant features in the signal without any domain knowledge [8] or hand-crafted features. Previous sleep stage studies [6,8] that built purely on CNN usually start with 1D convolutional layer followed by 1 to 5 consecutive convolutional layers (with ReLU and max pooling), fully connected layers and lastly a 5-class softmax to generate the final sleep stage prediction. Max pooling layers are used to prevent overfitting and to reduce computational cost.

Recurrent neural networks are powerful for sequential data. In the AASM scoring rules we can see that the scoring for a particular epoch can sometimes depend on the label for its previous epoch or its following epoch [8]. Therefore RNN has shown to be well-suited for discovering temporal dependencies in PSG data [6,7,10].

Frontiers in Computational Neuroscience [17] published an article with a very similar goal that led us down the right path and created inspiration for our CNN + LSTM model. They utilized three convolution layers, and three LSTM layers on top of the Keras python package. Instead of using a dense or fully connected layer at the end of their model, they were able to utilize the last LSTM layer to achieve the same effects. They utilized ADAM + Cross Entropy. They were able to achieve 73% accuracy with a training dataset of 19 subjects.

2. APPROACH AND IMPLEMENTATION

2.1 Dataset

For this project, we used sleep recording data from the Sleep-EDF Expanded database publicly available at PhysioNet[11, 12].

This database contains 197 samples of whole-night PSG recordings suitable for general purpose sleep analysis. All records contain readings from EEG, EOG, chin EMG and event markers with some samples containing respiration and temperature data.

The recordings are available in EDF/EDF+[13] format which is the de facto standard for PSG data. The data from these samples will be used to build the features that will be used to train our scoring models. Along with the sample data, there also is a set of expert-scored hypnograms available that contain ground truth annotations corresponding to each 30-second epoch in the PSG recordings. These patterns are W(Wake), R(REM), 1(N1), 2(N2), 3(N3), 4(N4), M (Movement time) and ? (not scored) [12].

These recordings were originally obtained from two separate studies [12, 14]. The first study contains 153 samples from healthy subjects not using sleep related medication. Each of these subjects has two PSG recordings of approximately 20 hours each. The EOG and EEG samples were taken at 100Hz, while the event markers were sampled at 1Hz. The second study contains data from 22 subjects who suffer from sleep issues. There are again two samples for each subject however this group contains one each of medicated and placebo dosed sleep assistance.

Other papers using this source data only used the first study of healthy individuals (known as the "sleep-cassette" study) and so we did the same. Three samples from that dataset were permanently lost from the source archives and are unavailable to us, which resulted in a total input dataset of 150 samples.

2.2 Input Data and Feature Selection

Typically EDF files containing PSG and Hypnogram data are manipulated using dedicated software to display and allow manual human annotation of the sleep recordings (such as Polyman, pictured in Figure 1). However, because the PSG/Hypnogram recordings are stored in the custom binary format, it is necessary to extract the readings into format consumable by the machine learning libraries used for the project. The software allows for exporting the raw sensor readings into a delimited text file format (tab-separated, tsv) and also, separately, can export the scoring data into a delimited format(comma-separated, csv).

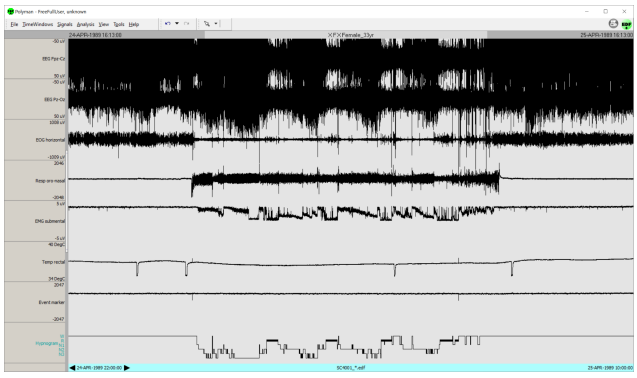

Figure 1: Example of a typical PSG/Annotation display

While this created data usable programmatically, there are three significant issues that need to be addressed. First, the size of the data is very large. From this dataset, the export of a single subject's raw sensor readings sampled at 100Hz resulted in a file approximately 400MB on average. This results in a raw input data size of nearly 60GB of raw data. The second issue is that each reading is individually captured, and there is a significant amount of processing required to then group each 1/100 of a second of raw sensor reading into the correct thirty second epoch, and then combine those with the human annotations to generate the feature vectors suitable for our model. Finally, we also noticed that each sample in our dataset included a considerable amount of awake time surrounding a relatively much smaller sleep time. To reduce the effect of having the awake epochs dominate the model training, we only included awake readings that were within 30 minutes prior to and after the sleep portion.

Because of the size and complexity of the work required to perform the necessary cleaning and aggregations on the raw source data, this preprocessing step would be very challenging to do on a typical consumer-grade laptop in a responsive and timely manner. We also strived to develop a repeatable methodology to quickly create usable data in a flexible, generalized data model directly from any EDF files, not just the source chosen for this project in order to support experimenting with different types of models and further studies. As such, the industry standard data engineering platform *Apache Spark* was used as the mechanism with which the bulk of the intensive data processing would be done on the raw data exported from the EDF files. The implementation that was built was specifically designed to handle any number of input files and volume of data without any additional manual processing.

The exported sensor readings and hypnogram scores were uploaded to *Azure Data Lake Storage* and were then processed using *Azure Databricks*. The score files were used to generate thirty second epochs of time and a corresponding sleep stage label per subject/sample. The raw sensor readings that were collected during each epoch were then aggregated into a vector per epoch, resulting in a data model as seen in Figure 2.

timestamp	annotation	EEG_Fpz-Cz_uV	EEG_Pz-Oz_uV	EOG_horizontal_uV	subject
1989-04-25 00:13:30	W	[8.111355311355311, 17.488644688644687, 21.239...	[-1.6996336996336874, -1.891575091575793, -4....	[19.465445665544572, 44.59804639804644, 33.7565...	SC4001
1989-04-25 00:14:00	W	[-10.736996336996336, -11.393406593406594, -4....	[-1.2197802197802075, -1.6036630036629915, -2....	[-25.378998778998724, -47.062026862026805, -37...	SC4001
1989-04-25 00:14:30	W	[61.37435897435897, 38.68131868131868, 49.5589...	[2.9069597069597193, 10.104761904761915, 15.09...	[46.07643467643473, 70.22344322344328, 6.65274...	SC4001
1989-04-25 00:15:00	W	[27.803663003663, 24.146520146520142, 24.14652...	[-4.098901098901087, -0.06813186813185584, 0.1...	[36.71330891330896, 6.159951159951213, 79.0937...	SC4001
1989-04-25 00:15:30	W	[69.15750915750915, 74.59633699633699, 79.0036...	[4.3465201465201595, 2.043223443223455, -1.50...	[56.425152625152267, 71.70183150183156, 11.0879...	SC4001
...	...	...	...	...	...
1991-09-27 08:32:30	W	[49.2888888888889, 57.733333333333334, 64.31111...	[3.7772893772893683, 14.891575091575081, 15.36...	[132.23882783882783, 126.4161172161172, 132.23...	SC4822
1991-09-27 08:33:00	W	[-84.04444444444444, -85.73333333333333, -75.4...	[-6.8620268620268705, -16.926251526251534, -8....	[-74.95262515262515, -76.8935286935287, -77.37...	SC4822
1991-09-27 08:33:30	W	[4.044444444444451, 2.2666666666666737, 6.4444...	[1.2124542124542033, -3.822222222222231, 1.592...	[71.10036630036628, 83.71623931623931, 89.0537...	SC4822
1991-09-27 08:34:00	W	[-43.955555555555555, -63.422222222222221, -70.8...	[-13.226617826617835, -8.381929181929195, -3.82...	[-63.30720390720391, -91.93553113553112, -129....	SC4822
1991-09-27 08:34:30	W	[21.73333333333334, 21.555555555555556, 21.9111...	[14.606593406593397, 13.846642246642235, 9.381...	[-90.96507936507938, -82.71623931623932, -72.0...	SC4822

Figure 2: Sample of processed output data

Due to size of the data, and a desire to maintain the vector format for each epoch, standard delimited text file is not a good option, so the processed dataset was stored in Parquet file format. *Apache Parquet* is an open-source column oriented file format that very efficiently serializes large semi-structured data like in our model. It also works very well with Spark as the Spark output writer is heavily optimized to use Parquet format. Using a Spark cluster of five nodes of modest size (8 cores, 32GB ram), the total time to read, process and write the entire output dataset from the full 60GB of raw data was only five minutes, thirty-six seconds.

Parquet is also very well supported by the standard data science tools used in this project (Pandas, NumPy, PyTorch), and is much more efficiently read into memory than textual files. Furthur customizations to this general data model would then be very easy to apply to fine-tune the input data for the specific models in our implementation with minimal processing overhead.

Specifically, the processed dataset output from the Spark cluster had annotations for the sleep stages corresponding to awake, rem, N1, N2, N3, N4, eye movement and some unscored epochs. We further refined this input in the following ways:

- Combined N3 & N4 stages to N3.
- Removed the epochs related to movement stage.
- Removed the unscored epochs.
- Annotated the string values of sleep stages to numeric values. The epochs corresponding to sleep stages of R, N1, N2, N3 & W became 0, 1, 2, 3 & 4, respectively.

Sleep stages N3 and N4 were combined into N3 to align with new AASM scoring rules. We removed M (Movement time) and ? (not scored) epochs since they don't belong to any of the sleep stages defined by AASM. Raw EEG data is used without any further signal transformations.

The output dataset after implementing the above steps had a total of 193501 epochs, with 25511, 21323, 67606, 12833, 66228 epochs corresponding to sleep stages R, N1, N2, N3 & W.

Since we had already systematically removed the data of the awake stage except 30 mins before and after the sleep, we got a significantly class balanced dataset, as shown in Figure 3 below.

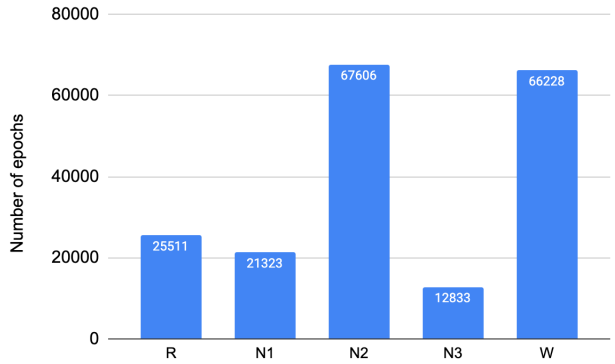

Figure 3: Sleep Stage Distribution after processing

The sleep signals corresponding to our selected channel appear pretty much normally distributed, as shown in Figure 4 below. This is a desirable state of the input data to reduce the possibility of bias and variance in the resultant data models.

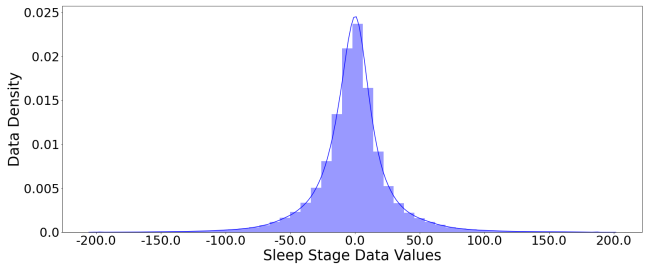

Figure 4: Bell Curve Shaped Sleep Signals in Input Dataset

2.3 Model Architecture

2.3.1 Non-Deep-Learning Architectures

We implemented Logistic Regression (LR), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA) for the input dataset. All 3000 features in a 30 seconds epoch for single channel data were used. We implemented standard LR, LDA and QDA algorithms so that we'll have enough comparison statistics to use against deep learning models.

2.3.2 CNN Architecture

Convolutional neural networks are used to extract signal patterns in each epoch. In our CNN models, The inputs are non-overlapping 30-second signal readings from three channels: EEG_Fpz-Cz, EEG_Pz-Oz, and EOG_horizontal. Given the sample rate of 100Hz on these channels, the input size is 3000 per epoch per channel. Since signals are one dimensional, we 1D convolution for all convolutional layers.

We experimented with two CNN models, one with a relatively large kernel size of 50 and stride size of 6 (model A), and the other with relatively small kernel size 8 and stride size of 1 (model B).

2.3.2.1 CNN Model A

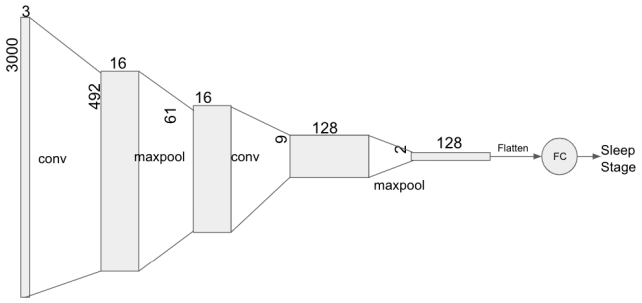

Figure 5: CNN architecture (model A)

Layer	Kernel size	# filters	Stride	Output Size
input				(3, 3000)
convolutional	50	16	6	(16, 492)
batch-norm, ReLU				(16, 492)
max-pooling	8		8	(16, 61)
dropout	p=0.5			(16, 61)
convolutional	8	128	8	(128, 9)
batch-norm, ReLU				(128, 9)
max-pooling	4		4	(128, 2)
fully-connected, ReLU				5
max				1

Table 1: CNN model parameters (model A)

As shown in Figure 5 above, CNN model A contains 2 convolutional layers, 2 max-pooling layers and 1 fully connected layer. Each convolutional layer performs a 1D-convolution, batch normalization, and rectified linear unit (ReLU) activation. A fully-connected layer then takes in the flattened representation and outputs 5 numbers corresponding to the logits of the probability that an epoch belongs to a category. The category with the maximum value is taken to output the final sleep stage.

Due to the kernel size and stride size, the dimension of the data reduces quickly after 2 convolutional layers. So we didn't apply a third layer of convolution to this model.

2.3.2.2 CNN Model B

CNN model B consists of similar components. The main difference is a smaller kernel and stride size. Figure 6 illustrates model B with three convolutional layers. We've also experimented with using just two convolutional layers by dropping the last convolution operation, to compare results with model A. Similarly a fully-connected layer is used to consolidate vectors to 5 categories.

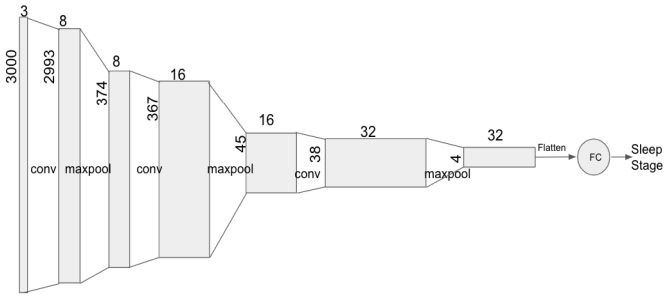

Figure 6: CNN architecture (model B) with three convolutional layers

Layer	Kernel size	# filters	Stride	Output Size
input				(3, 3000)
convolutional	8	8	1	(8, 2993)
batch-norm, ReLU				(8, 2993)
max-pooling	8		8	(8, 374)
convolutional	8	16	1	(16, 367)
batch-norm, ReLU				(16, 367)
max-pooling	8		8	(16, 45)
convolutional	8	32	1	(32, 38)
batch-norm, ReLU				(32, 38)
max-pooling	8		8	(32, 4)
fully-connected, ReLU				5
max				1

Table 2: CNN model parameters (model B) with three convolutional layers

2.3.2.3 CNN Model Training and evaluation

Models are trained with Adam optimizer with a learning rate of 0.001. We use cross entropy loss as the loss function.

Based on some preliminary studies, we found that CNN models perform better when class labels are balanced. Therefore for all CNN model training and evaluation, we employed further class-balanced random sampling so that each class has the same number of epochs. Since N3 has the least number of epochs, 12833, we resampled all the other classes to have 12833 epochs.

The 64165 epochs after resampling are then randomly split into 70-30 for training and testing.

2.3.3 CNN-RNN Architecture

RNN and more specifically Long short-term memory (LSTM) is a popular architecture among sleep stage classification papers due to its ability to learn temporal relationships between epochs [6,7,10].

In the CNN-RNN model, we also used non-overlapping 30-second signal readings from three channels: EEG_Fpz-Cz, EEG_Pz-Oz, and EOG_horizontal. We first use CNN to learn a local representation Φ_t of each epoch. The hyperparameters for CNN are the same as that in CNN model B shown in Table 2.

$$\Phi_t = \text{CNN}(\text{epoch } t)$$

where t = 1,2, ...n. This representation is flattened to a 1-D vector of length 128. Since each sleep recording consists of a different number of epochs, we padded the sequence to match the maximum number of epochs in all sleeps, which is 2795, to allow batch processing.

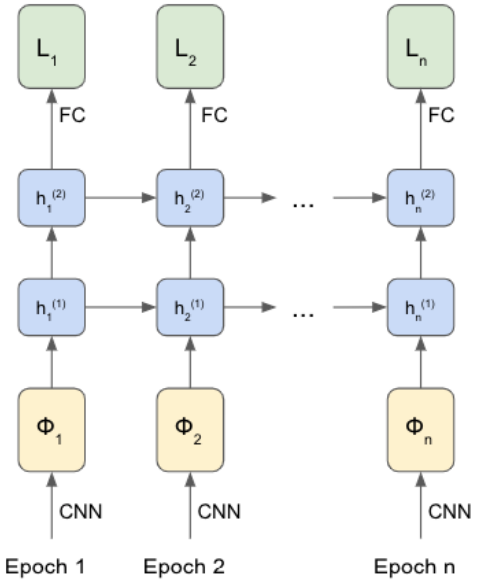

Figure 7: CNN-RNN architecture

The padded sequence $[\Phi_1, \Phi_2, \ldots \Phi_{2795}]$ from each sleep is then passed to a stacked LSTM layer to learn the temporal dependencies. We stacked two LSTMs together, with the second LSTM taking the outputs of the first to compute the result. In this stacked LSTM layer, we have experimented with different numbers of hidden units.

Lastly, the hidden state from each epoch is passed through a fully connected layer to predict the final sleep stage. Figure 7 illustrates this many to many RNN structure. The padded epochs are removed before passing to the loss function in training, and also before evaluation.

The model is trained with Adam optimizer with a learning rate of 0.001. We use cross entropy loss as the loss function.

For CNN-RNN model training and evaluation, we used all the epochs in spite of the class imbalance. This is because RNN needs consecutive epochs to learn temporal relations. So we decided to feed in the entire sequence of epochs. We split the recordings

70-30 for training and evaluation. There are in total 150 recordings. We used 105 for training and 45 for testing.

2.4 Model Evaluation Metrics

Evaluation is done by splitting the data into training and validation sets as described above.

Classification performance of our models is evaluated with accuracy, average F1 score, per-class precision, recall, F1 score and Cohen's Kappa score. F1 score is the harmonic mean of precision and recall. F1 score is a more comprehensive measure since precision and recall can increase at the cost of the other. Cohen's Kappa score measures inter-rater reliability. It allows us to tell to what extent the labels produced by our model are in agreement with that from a human expert. The overall score from eight European centers is 0.76 [5], and we will be comparing ours with this human level performance.

2.5 Model Implementation

We implemented our model using PyTorch 1.3.1 and generated the evaluation metrics using scikit learn 0.23.

3. RESULTS

We've summarized below the accuracy score of all the algorithms we've applied on the sleep data:

	Logistic Regression	LDA	QDA	CNN	CNN-RNN
Accuracy Score	0.29	0.37	0.60	0.77	0.79
Average F1	0.21	0.19	0.32	0.77	0.72
Cohen's Kappa	0.02	0.06	0.41	0.71	0.71

Table 3: Evaluation scores from different models

3.1 QDA

The confusion matrix for QDA doesn't look ideal. It has a lot of big values on off diagonal cells. It will, however, serve as an effective non-deep learning model to compare against deep learning models.

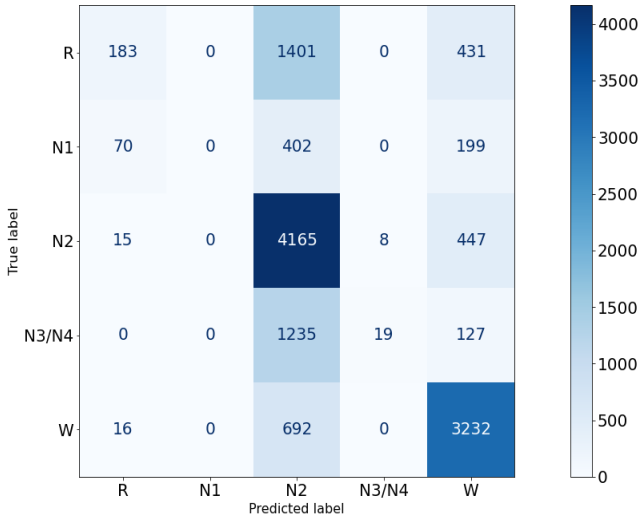

Figure 8: Confusion matrix from validation set for QDA model

3.2 CNN

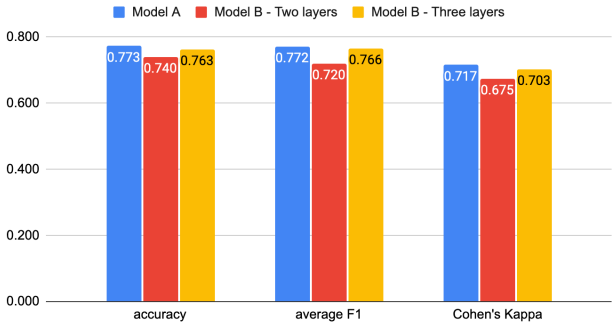

Figure 9: Average accuracy, F1 and Cohen's Kappa score for CNN models

As is shown in Figure 9, CNN model A and model B with three convolutional layers have relatively close model performance. Model B with three layers performs better than that with two convolutional layers.

3.2.1 CNN Model A

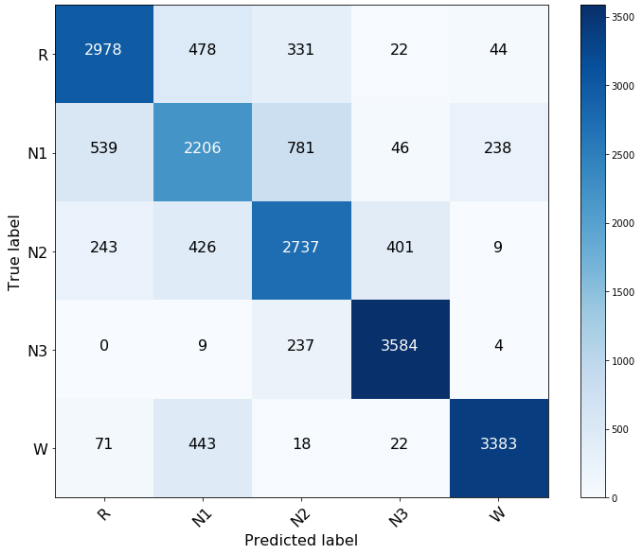

Figure 10: CNN model A confusion matrix from validation set

	R	N1	N2	N3	W
precision	0.777	0.619	0.667	0.880	0.920
recall	0.773	0.579	0.717	0.935	0.859
F1	0.775	0.598	0.691	0.906	0.889

Table 4: CNN model A per-class precision, recall and F1 score

CNN model A performs the best on classifying N3 and W stages, with 0.91 and 0.89 per class F1 score, respectively. The most misclassified class is N1. The largest off diagonal numbers come from pairs R-N1, N1-N2, followed by R-N2. The remaining pairs have a very small chance of being misclassified to each other.

3.2.2 CNN Model B

Shown below, are the results from model B with three convolutional layers.

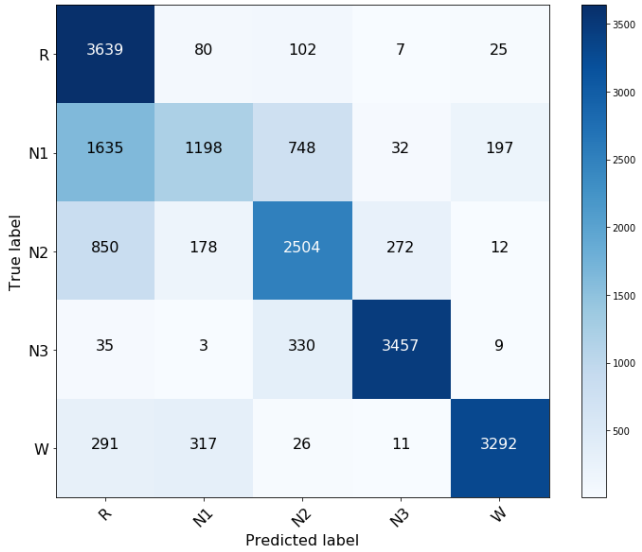

Figure 11: CNN model B confusion matrix

	R	N1	N2	N3	W
precision	0.723	0.578	0.687	0.929	0.952
recall	0.827	0.629	0.677	0.870	0.808
F1	0.772	0.602	0.682	0.898	0.874

Table 5: CNN model B per-class precision, recall and F1 score

Similar to model A, It performs the best on classifying N3 and W stages. The most misclassified classes are N1 and N2. Considerable amounts of N1 epochs are classified as R or N2.

3.3 CNN-RNN

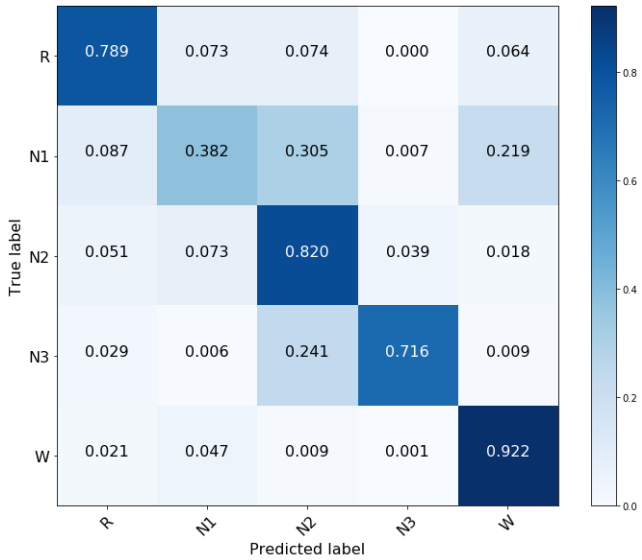

Figure 12: normalized confusion matrix for CNN-RNN model

Figure 12 shows the confusion matrix from the baseline model with hidden size of 32 and 2 layers of LSTM stacked together. Due to class imbalance, we normalized the confusion matrix over the rows (i.e. true labels) for better visualization.

	R	N1	N2	N3	W
precision	0.717	0.462	0.815	0.755	0.901
recall	0.789	0.382	0.820	0.716	0.922
F1	0.751	0.418	0.817	0.735	0.911

Table 6: CNN-RNN model per-class precision, recall and F1 score

We experimented with different hidden sizes while keeping the number of layers at 2. The result is shown in Figure 13.

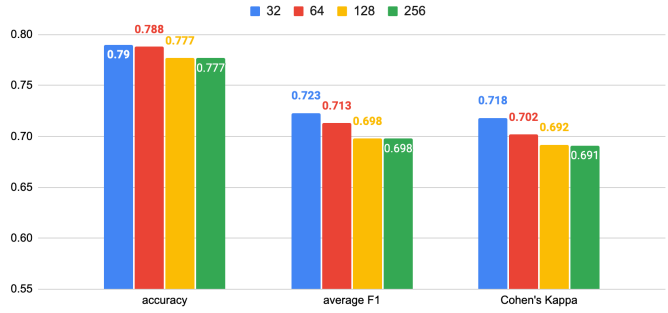

Figure 13: Influence of hidden size on accuracy, average F1 score and Cohen's Kappa score

We can see that model performance decreases slightly as hidden size increases.

4. EXPERIMENTAL EVALUATION

4.1 Non-Deep-Learning Models

Since the input sleep signals have non-linear patterns, it was evident that nonlinear algorithms will result in better accuracy. As we've seen, QDA has the highest accuracy. Logistic regression, which is based on linear decision boundary, performs worse than the baseline accuracy. QDA improves almost 100% over LDA, in terms of accuracy. Even with the improved performance in QDA, some sleep stage classes are misclassified in the confusion matrix. We hope to get better F1 scores for these classes, while using deep-learning algorithms.

4.2 CNN Model Performance

CNN model A and model B with three convolutional layers have achieved comparable performance. Model A achieved accuracy of 0.77 and average F1 of 0.77. Previous research [8,15] results reported accuracy in the range of 0.74 to 0.78, F1 of 0.79 and Cohen's Kappa of 0.7. While these results are not directly comparable due to the differences between the datasets, hyperparameters, or possible addition of hand-engineered features, it showed that our model performance is in a good and reasonable range.

In model A, the filter size of 50 in the first convolutional layer seems to capture spatial information better than that of size 8 in

model B. But with one more layer added, model B is able to achieve similar performance.

Regardless of the filter size, N1 is always the most difficult class to learn among the 5 classes. This is consistent with the result from another published paper [8]. We can rule out the influence of class distribution of sleep stages because we did class-balanced resampling. The possible explanation for the low N1 performance could be that the AASM scoring rules allows N1 under certain conditions to be classified as N2 as well, as pointed out in [7]. Further understanding in the AASM rules and human scorer experiences could be helpful in adding some distinguishing features to help with model performance.

4.3 CNN-RNN Model Performance

Our CNN-RNN model achieved an accuracy of 0.79, with an average F1 score of 0.72. The Cohen's kappa coefficient is 0.71, which shows that the labels produced by our model are in substantial agreement with that from a human expert.

Since there are no guidelines on determining the hidden size in an LSTM, we experimented with hidden sizes of 32, 64, 128 and 256. We found that increasing the hidden size didn't result in a better overall performance. In terms of per class F1 score, there's no hidden state setting that leads to a better prediction for a specific class.

In the CNN-RNN model, the N1 stage has the lowest F1 score among all five classes. This is consistent with previous studies [10, 17] that use a CNN-RNN approach. It could be because N1 stage is under-represented, and that the scoring rule is favoring N2 as mentioned earlier.

5. CONCLUSION

The recent advances in machine learning combined with statistical learning can help distinguish the sleep stages successfully and effortlessly. The automated sleep scoring mechanism discussed in this paper can reduce the burden on clinicians, contribute to the in-home device implementation of a sleep monitoring system and benefit the sleep research community[16].

Our work is based on PSG data from 150 samples. We experimented with LR, LDA, QDA, CNN and CNN-RNN machine learning models. Deep learning models outperform traditional machine learning methods. CNN models have shown promising results and CNN-RNN model shows the best result on sleep stage classification.

There is abundance of scope for further research in this area by adding more data for the study and supplementing the sleep signals with other diagnosis and medical records to discover the sleep related health problems and possible cures. Additional possible avenues of improvement include shortening the sequence length to about 90 minutes, experimenting with LSTM with deeper layers, and using a different dataset for performance testing to add robustness to the model.

6. CHALLENGES AND LEARNINGS

The biggest challenge we faced in this project is the implementation of CNN-RNN model. Since CNN and LSTM layers are expecting different arrangements of epochs, i.e individually and in a sequence. We were able to overcome it by reading in the data into sequences of epochs for each sleep, and reshape it to remove the sequence dimension for CNN.

Originally trying to implement RNN without the CNN component led to very low performance, the use of greater computational resources, and the epochs took a lot longer to train. This was fixed by implementing the three CNN layers above RNN as explained above.

In the model evaluation phase for CNN only models, we are challenged by the nature of asymmetric distribution of the sleep stages. Although a lot of the W stages are excluded by removing W epochs that are more than 30 minutes before and after the sleep period, there still exists an non-negligible imbalance between the other stages. We tried using a weight parameter in the cross entropy loss function to give the mislabeled class more weights, but the results show that the increase of F1 in one class comes at the price of other classes. Therefore we finally decided to resample every class to have the exact same amount of epoch, which achieved the best result.

Another learning is that it is important to design the experiments beforehand. We found ourselves trying out various hyperparameters blindly at first without a coherent story, and that led to a lot of wasted runs.

7. CONTRIBUTIONS

All group members contributed equally to this project. Common tasks such as project choice, scope, literature review, report/presentation completion etc were carried out jointly by everyone. While all team members contributed throughout all areas of the project, specific areas of focus were led by individual teammates as follows:

- Jason Kolter: Source data acquisition and cleaning/processing for model training and testing
- Pradeep Dwivedi: Data exploration and traditional non-DL model implementation for baseline analysis
- Zichun Xu and Michael Renteria: CNN and RNN deep learning model implementation and analysis

8. REFERENCES

[1] Raymond C. Rosen, Mark Rosekind, Craig Rosevear, Wesley E. Cole, William C. Dement, Physician Education in Sleep and Sleep Disorders: A National Survey of U.S. Medical Schools, Sleep, Volume 16, Issue 3, May 1993, Pages 249–254, https://doi.org/10.1093/sleep/16.3.249

[2] Abad, V. C., & Guilleminault, C. (2003). Diagnosis and treatment of sleep disorders: a brief review for clinicians. Dialogues in clinical neuroscience, 5(4), 371–388.

[3] Iber, C., S. Ancoli-Israel, A. Chesson, and S. F. Quan. The AASM Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and Technical Specifications. Westchester, IL:American Academy of Sleep Medicine, 2007.

[4] Miladinović Đ, Muheim C, Bauer S, Spinnler A, Noain D, Bandarabadi M, et al. (2019) SPINDLE: End-to-end learning from EEG/EMG to extrapolate animal sleep scoring across experimental settings, labs and species. PLoS Comput Biol 15(4):e1006968. https://doi.org/10.1371/journal.pcbi.1006968

[5] Danker-Hopfe H, et al. Interrater reliability for sleep scoring according to the Rechtschaffen & Kales and the new AASM standard. J Sleep Res. 2009;18(1):74–84.

[6] S. Biswal, J. Kulas, H. Sun, B. Goparaju, M. Brandon Westover, M. T. Bianchi, and J. Sun. SLEEPNET: Automated sleep staging system via deep learning. 26 July 2017.

[7] L. Zhang, D. Fabbri, R. Upender, and D. Kent. Automated sleep stage scoring of the sleep heart health study using deep neural networks. 2019.

[8] O. Tsinalis, P. M. Matthews, Y. Guo, and S. Zafeiriou. Automatic sleep stage scoring with single-channel eeg using convolutional neural networks. arXiv preprint arXiv:1610.01683, 2016.

[9] Kiranyaz, S. (1), Ince, T. (2), Abdeljaber, O. (3), Avci, O. (3), & Gabbouj, M. (4). (n.d.). 1-D Convolutional Neural Networks for Signal Processing Applications. ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 2019–May, 8360–8364.

[10] A. Supratak, H. Dong, C. Wu and Y. Guo, "DeepSleepNet: A Model for Automatic Sleep Stage Scoring Based on Raw Single-Channel EEG," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 25, no. 11, pp. 1998-2008, Nov. 2017, doi: 10.1109/TNSRE.2017.2721116.

[11] Goldberger, A., et al. "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online]. 101 (23), pp. e215–e220." (2000).

[12] B Kemp, AH Zwinderman, B Tuk, HAC Kamphuisen, JJL Oberyé. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave micro-continuity of the EEG. IEEE-BME 47(9):1185–1194 (2000)

[13] B Kemp, J Olivan. European data format 'plus' (EDF+), an EDF alike standard format for the exchange of physiological data. Clinical Neurophysiology 114:1755–1761 (2002).

[14] MS Mourtazaev, B Kemp, AH Zwinderman, HAC Kamphuisen. Age and gender affect different characteristics of slow waves in the sleep EEG. Sleep 18(7):557–564 (1995).

[15] I. Al-Hussaini, C. Xiao, M. B. Westover, and J. Sun. Sleeper: inter- pretable sleep staging via prototypes from expert rules. In Machine Learning for Healthcare Conference, pages 721–739, 2019.

[16] Md Mosheyur Rahman, Mohammed Imamul Hassan Bhuiyan, Ahnaf Rashik Hassan: Sleep stage classification using single-channel EOG:Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, Dhaka, 1205, Bangladesh: https://doi.org/10.1016/j.compbiomed.2018.08.022.

[17] Bresch Erik, Großekathöfer Ulf, Garcia-Molina Gary: Recurrent Deep Neural Networks for Real-Time Sleep Stage Classification From Single Channel EEG:Frontiers in Computational Neuroscience : https://www.frontiersin.org/article/10.3389/fncom.2018.00085