Automated Sleep Stage Scoring On Single Channel Signals With Deep Learning Approach

Ruf Théodore,
Double Degree in Smart Systems, École de Technologie Supérieure, Montréal, Canada

Abstract — **Electroencephalogram (EEG) is a common base signal used to monitor brain activity captured through electrodes placed on the human scalp. In this paper, we propose an automatic sleep stage annotation method using a single-channel EEG signal. The Model is composed of a deep convolutional neural network (CNN) to extract time-invariant features. In addition, to reduce the effect of the class imbalance problem existing in the available sleep datasets, we applied class weighting. We evaluated the proposed method on different single-EEG channels (i.e., Fpz-Cz and Pz-Oz) from the Physionet Sleep-EDF datasets published in 2013 and 2018. The evaluation results demonstrate that the proposed method achieved an overall accuracy of 83.7%, and a macro F1-score of 60.17%.**

Keywords* — *Sleep stage scoring, EEG analysis, deep learning, Convolutional Neural Networks.

I. Introduction (*Heading 1*)

In this project, we work with the 2018 Physionnet Sleep-EDF dataset which are 153 whole-night PolySomnoGraphic sleep recordings of ~20 hours containing electroencephalogram (EEG), electrooculogram (EOG), and electromyogram (EMG) signals. These signals are widely used for sleep disorders diagnosis (e.g., sleep apnea, parasomnias, and hypersomnia) which are essential in improving individuals' overall health and well-being. As the manual interpretation of EEG signals for sleep stage classification is a labor-intensive and time-consuming task, our goal here is to develop a deep learning model for sleep stage classification.

This model is based on two paperwork; the first one [1] introduces an automatic sleep stage annotation method named SleepEEGNet. It employs deep convolutional neural networks (CNNs) and a sequence-to-sequence model (combination of attention and bidirectional Recurrent Neural Networks). This method provides high level of accuracy in sleep stage annotation and has the potential to assist sleep specialists in making more accurate diagnoses. The second paper [2] focuses on only using convolutional neural networks (CNNs). The authors conducted an analysis of the learned filters within the CNN and suggested that the CNN, without prior domain knowledge, autonomously learned to distinguish among different normal sleep stages.

My approach is based on the methodology of the first paper, it is not entirely alike but somehow following the same process. Although, I couldn't reproduce it entirely, this project provides useful insights regarding the difficulties of applying complex research methods. Which contributes to understanding how deep learning concepts in the course relate to real-world challenges and the importance of adapting them effectively.

II. Methodology

A. *Signal pre-processing and preparation*

The first step was to split the wanted channels “EEG Pz-Oz” or “EEG Fpz-Cz” from the sleep recordings into 30 seconds samples (epochs) as this duration aligns with the expert’s annotation of sleep stages.

As the amplitude values in the raw signals were very low (~10e-6 order of magnitude), it was necessary to normalize the data. The signals were recentered around zero (Zero Mean) and put on the same scale (Unit Variance) to make the data more suitable for training deep learning models. Basically, it is done by substracting each sampled point by the sequence mean value and dividing this result by the standard deviation value calculated on the whole sequence. This step is crucial when working with scale-sensitive algorithms, such as neural networks. It allows the model to be more robust to amplitude variations in EEG signals.

From there, I built a data frame containing the *n* normalized epochs (rows) with their corresponding 3000 sampled points and label (columns). Each 30 seconds epoch contains 3000 sampled points because the EEG sampling frequency is 100Hz.

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{3000,1} & x_{3000,2} & \cdots & x_{3000,N} \end{bmatrix}$$

Fig. 1. $X_{3000,N}$ Input matrix.

From this data frame, it was easy to remove the unwanted samples such as the unannotated and movement time ones, and to do labels one hot encoding which is turning the categorical labels into a binary class matrix.

Finaly, the data was randomly split in three sets: Training (70%), validation (10%) and test (20%).

B. *Feature engineering*

After doing some data visualization, I noticed that the plotted signals were a bit noisy and showing some trends. To rectify that, I created a band pass filter [0.5-35Hz] to remove powerline noise, focus on relevant frequencies and improve the overall quality of the signal.

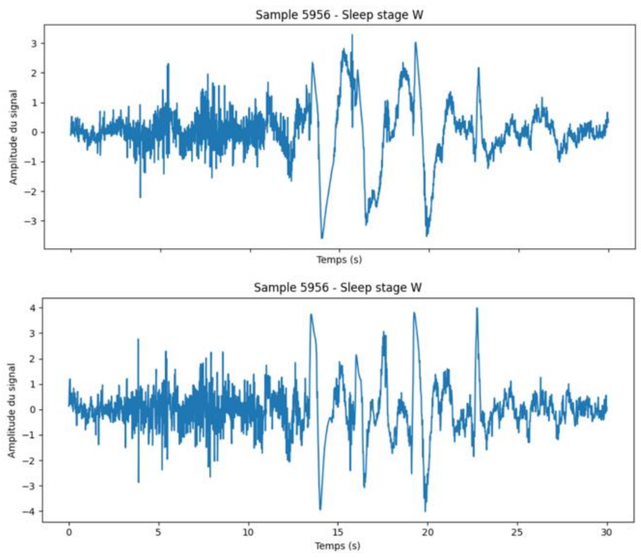


Fig. 2. Random "W" Labeled epoch before and after applying the band-pass filter.

We get cleaner signals which is generally better for the model to learn.

C. *Deep learning model*

Here is where I had some issues with reproducing the paper's model which is the combination of a CNN and sequence-to-sequence with attention model. As it was complicated and briefly described in the article, I decided to start with a simple CNN architecture and to improve it through model validation that is described in section D.

After evaluating my model with different hyperparameters (Number of layers, learning rates etc…), I ended up with the following neural network architecture. We have an input layer taking (3000,1) shape inputs, a 1D convolutional layer with ReLU activation function, a max pooling layer with a pool size of 2, a flattening layer to prepare the data for a fully connected dense layer with ReLU activation function and finally an output layer wich is a dense layer with a Softmax activation for multiclass classification.

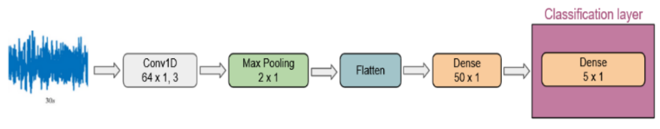


Fig. 3. CNN architecture used in the approach.

D. *Validation configuration*

Model validation was mainly about tuning the model to overcome overfitting and improve model generalization. Here are the main validation configuration that were made:

- Regularization method such as early stopping (with patience 3) on validation set. At a number n of training epoch, if the three next loss validation values have not increased then training stops.
- Considering more sleep recordings, which basically is considering more 30 seconds epochs.
- Implementing class weighting as the data shows an important imbalance between the classes.

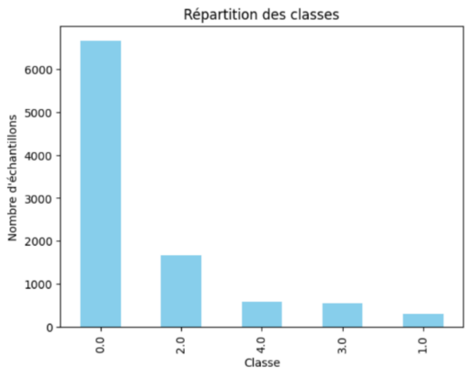
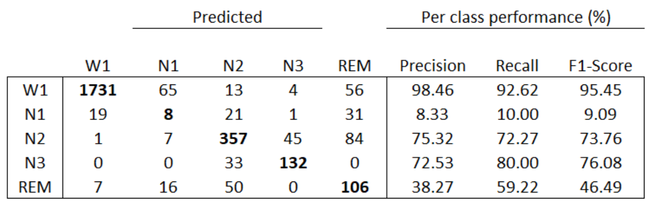


Fig. 4. Density plot of the selected epochs to visualize the data distribution.

III. RESULTS

My model reached an overall accuracy of 83.7% on test set (data the CNN never saw), but at some point, showed some difficulties to learn the signal variations between some classes. So, to provide a more detailed and nuanced understanding of how well my sleep stage classification model is performing, I used a confusion matrix. It is widely used, especially in the context of imbalanced datasets. Furthermore, it gives the possibility to compare results with the literature with precise metrics such as Precision, Recall and F1-score.

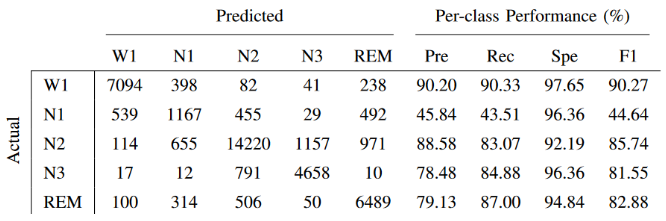
TABLE I. CONFUSION MATRIX AND PER-CLASS PERFORMANCE ACHIEVED BY OUR MODEL USING PZ-OZ EEG CHANNEL OF THE EDF-SLEEP-2018 DATABASE.



	Predicted					Per class performance (%)		
	W1	N1	N2	N3	REM	Precision	Recall	F1-Score
W1	**1731**	65	13	4	56	98.46	92.62	95.45
N1	19	**8**	21	1	31	8.33	10.00	9.09
N2	1	7	**357**	45	84	75.32	72.27	73.76
N3	0	0	33	**132**	0	72.53	80.00	76.08
REM	7	16	50	0	**106**	38.27	59.22	46.49

In comparison to SleepEEGNet [1], we can say that our results are consistent. Of course, our predictions are globally less correct.

TABLE II. CONFUSION MATRIX AND PER-CLASS PERFORMANCE ACHIEVED BY SLEEPEEGNET USING PZ-OZ EEG CHANNEL OF THE EDF-SLEEP- 2013 DATABASE.



		Predicted					Per-class Performance (%)			
		W1	N1	N2	N3	REM	Pre	Rec	Spe	F1
Actual	W1	7094	398	82	41	238	90.20	90.33	97.65	90.27
	N1	539	1167	455	29	492	45.84	43.51	96.36	44.64
	N2	114	655	14220	1157	971	88.58	83.07	92.19	85.74
	N3	17	12	791	4658	10	78.48	84.88	96.36	81.55
	REM	100	314	506	50	6489	79.13	87.00	94.84	82.88

For REM class we get a bit less than a half that is correctly predicted while SleepEEGNet seems to classify very well.

For N1 we notice that almost all of our samples are not well predicted. When looking at SleepEEGNet performances for this class, it seems like it also has some issues with learning it.

There can me many reasons leading to our less good results, it is what will be discussed in the next section.

IV. Discussions

Before going straight forward to the discussions, here is a visualization of my loss and accuracy curves on training and validations sets.

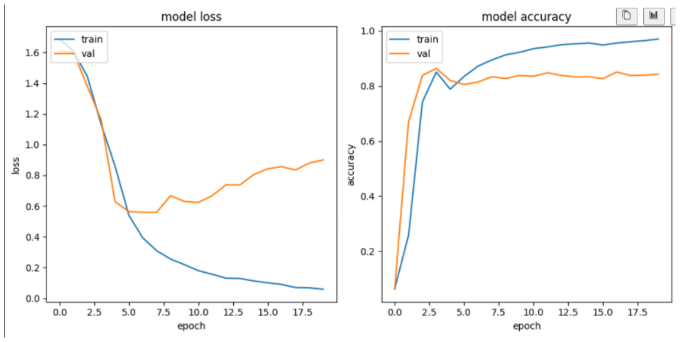


Fig. 5. Loss and accuracy curves on the train and validation set.

We can notice straight away that without the early stopping configuration, the model would have overfitted.

The first thing to be mentioned is the huge difference between the paper's neural network and mine. In my model, I did not use any Reccurent Neural Network (RNNs) and not having RNNs means not having the possibility to capture the complex and long short-term context dependencies between sleep epochs and scores during training.

Also, as I went from scratch, there are many steps I reproduced with my own methods and many others I had to skip due to lack of time and computational limitations.

Here are some implementations that would probably increase my model performances:

- Using all PSGs, as I only used 5 out of the 153 available.
- Using k fold cross validation to get more representative results.
- Implementing Novel Loss Function [1] to deal with class imbalance when backpropagating.
- Trying SMOTE technique which is also dealing with class imbalance but with some sort of data augmentation.
- Trying L2 regularization or any other complex techniques to overcome with overfitting.

The above steps could probably help to cope with the class imbalance issue but more specifically the stage N1, that is more difficult than any other sleep stages to score. I would say that it is the priority task for any future work on this automated sleep stage scoring algorithm.

V. Conclusion

Although my model is not performing as well as any other literature, I presented an algorithm for automated sleep stage annotation problem. The proposed method demonstrates the ability of convolutional neural network to learn to distinguish among different normal sleep stages efficiently. Table 1 shows that our model is able to learn some signal variation which could be assumed to be a good start towards being a significantly performing model for sleep stage scoring using single channel electroencephalogram signals.

References

[1] Mousavi, Sajad, Fatemeh Afghah, et U. Rajendra Acharya. " SleepEEGNet: Automated Sleep Stage Scoring with Sequence to Sequence Deep Learning Approach ". PLOS ONE 14, n? 5 (7 mai 2019): e0216456. https://doi.org/10.1371/journal.pone.0216456

[2] Tsinalis, Orestis, Paul M. Matthews, Yike Guo, et Stefanos Zafeiriou. « Automatic Sleep Stage Scoring with Single-Channel EEG Using Convolutional Neural Networks ». arXiv, 5 octobre 2016. https://doi.org/10.48550/arXiv.1610.01683.

[3] « 20230802.SleepEdfDataset-SleepStageClassify ». Consulté le 3 décembre 2023. https://kaggle.com/code/bobaaayoung/20230802-sleepedfdataset-sleepstageclassify.

[4] « Papers with Code - BiLSTM Explained ». Consulté le 3 décembre 2023. https://paperswithcode.com/method/bilstm.

[5] Course by Mohamad Forouzanfar, MACHINE LEARNING FOR THE ANALYSIS OF BIOMEDICAL SIGNALS, Autumn session.

[6] Labs by Boshra Khajehpiri, Autumn session.

[7] Mishra, Aditya. « Metrics to Evaluate Your Machine Learning Algorithm ». Medium, 28 mai 2020. https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234.

[8] Mishra, Aditya. « Metrics to Evaluate Your Machine Learning Algorithm ». Medium, 28 mai 2020. https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234.