Automated Classification of Sleep Stages using Multi-head CNN and LSTM

Emma Yu
Queen's University
EmmaYu.cs@gmail.com

Abstract

Sleep is crucial to human health and identification of sleep stages can be used as indicators for diagnosis of many sleep disorders. The traditional sleep stage classification that relies heavily on physicians' inspection is expensive and error prone. To mitigate the bottleneck of manual sleep data analysis, this project presents a hybrid multi-head conventional neural networks (CNN) with Long-Short Term Memory (LSTM) method to automatically classify the sleep stages based on single channel electroencephalograph (EEG) data which is in time series format. Three models are built step by step with the same parameters used for the common characteristics in each model. Here, loss and accuracy are used to evaluate the performance of the neural network. The experimental results show that multi-head CNN model can achieve a good accuracy above 73.9% even with only 3 parallel heads and 2 fully connected layers. With only one additional LSTM layer stacked on top of multi-head CNN, the performance improved drastically from 73.9% to 82.143%.

Keywords

Neural Network, Sleep staging, Classification

I. INTRODUCTION

Sleep is fundamental to human health. People suffering inadequate sleep are at a heightened risk of obesity, diabetes, and high blood pressure. Identification of sleep stages is playing a pivotal role in analysis of sleep quality and diagnosis of many sleep disorder related pathologies, such as insomnia, narcolepsy, and dementia. As stated in American Academy of Sleep Medicine manual, sleep can be segmented into five stages: Wake (W), Non-Rapid Eye Movement stages N1 (for drowsiness or transitional sleep), N2 (for light sleep), and N3 (for deep sleep), and Rapid Eye Movement (REM), which can be identified by analyzing polysomnograms (PSG) data of patients during sleep. PSG data is a collection of data gathered from attached electrodes and other sensors, including electroencephalography (EEG) acquiring brain activities, electrooculography (EOG) capturing eye movements, electromyography (EMG) representing muscle activities and other related signals, among which EEG is most widely used.

Traditionally, sleep stage classification relies largely on manual inspection of PSG data by well-trained physicians or specialists, which is time consuming and subject to human errors. Thus, there is a significant need to efficiently identify sleep stages and ensure its high reliability. There are a multitude of methods automating the sleep stage classifications. Admittedly, most of those state-of-the art sleep stage classifiers have declared good accuracy, but their performance strongly depends on the complexity of the models, in which case the cost will grow exponentially with the dozens of hundreds of neurons in a very wide or deep network.

This project is levelled at developing a sleep stage classification system with a few number of network parameters and layers while still having acceptable classification accuracy, based on the public Sleep-EDF database [5]. It is been proved that deep learning-based methods that directly use multi-channel data as input have not effectively improved classification accuracy and that EEG signal is among the most representative signals for sleep staging analysis. In addition, the simple configuration of single EEG channel data facilitates continuous monitoring at home with wearable EEG acquiring devices. In this project, single channel EEG data will be used in distinguishing sleep stages.

CNN has demonstrated its ability serving as classifiers in a variety of domains. While it is widely used in image processing, this project takes a challenge and applies CNN in time series data. Intrigued by the multi-head attention mechanism in the Transformer, I will apply 3 head CNN to capture intrinsic signal characteristics with various time scales and explore CNN capabilities in classifying sleep stages. Further, to improve the performance of the 3-head CNN model, one more LSTM layer was stacked afterwards and the accuracy improved drastically from 73.9% to 82.143%.

The rest of this project report is organized as follows. Section II discusses the related work. Section III provides an overview of this project. The experiments conducted on detailed models are described and results are analysed in Section IV. Finally, Section V reports conclusions as well as the future work that can be done based on the current study.

II. LITERATURE REVIEW

There exists a decent body of literature on classifying sleep stages using machine learning (ML) and/or deep learning (DL) methods with analysis of sleep monitoring data from PSG. A recent study [2] utilized support vector machine (SVM) in sleep stage classification and proves good accuracy of 91.1%. However, data has to be denoised and as many as 41 features has to be extracted in their study. Another study [3] incorporates difference visibility graphs to SVM in order to further improve the classification accuracy by 3.8% with 9 features processed.

Convolutional neural networks (CNN) have been exploited for both automated feature construction and classification. CNN

is robust in exploring feature sets by convolving multiple filters with small segments of input data to extract time-invariant features. Authors of [1] conducted experiments with EEG & EOG datasets using 6 convolutional layers and reported an accuracy of 83.7%. While CNN is powerful in extracting time-invariant features, it fails to capture temporal features. Yulita et al. [4] proposed a model architecture that utilizes deep belief network to extract 28 features and long short-term memory network for classification based on EEG, EMG and EOG data. 86.12% accuracy was reported in their study.

These studies demonstrate that it is possible to identify sleep stages exploiting ML and/or DL techniques with a reasonably good performance. However, the performance hinges heavily on the complexity of models. Either huge datasets has to be used or a large number of features require extraction. Whether those automatic classifiers are practical enough to be deployed on wearable devices is until now an open question. Currently there is no single model that performs best in all classes and for all datasets. What I seek in this project is to develop an automatic sleep stage classification system with moderate complexity and able to be generalized to other datasets.

III. Approach

The approach used for sleep staging is improved step by step and the backbones used are CNN and LSTM. CNN has demonstrated its ability serving as classifiers in a variety of domains. While it is widely used in image processing, this project takes a challenge and applies CNN in time series data. Initially, this project starts from a simple CNN with only 1 head to explore its capability in sleep staging. Confronted with the unsatisfactory accuracy and intrigued by the multi-head attention mechanism in the Transformer, I will apply 3 head CNN to capture intrinsic signal characteristics with various time scales and evaluate the performance. It is acknowledged that LSTM works well with sequence. To further improve the classifying performance of the 3-head CNN model, one more LSTM layer is stacked afterwards.

The design architecture for the proposed approach is shown in Fig.1. All the three parallel CNN heads receive the exactly same input but have distinguished kernel sizes of 3, 5, and 11 respectively. In order to mitigate the overfitting problem, drop-out strategy is utilized right after the CNN layer. Given that max-pooling is capable of extracting sharp characteristics and reducing computations, it is also be placed after the dropout layer. All the results from the 3 routes are then be concatenated together as an effort to produce better results. An additional LSTM layer are integrated into this architecture following the concatenation process. Finally, The model is then connected to a dense layer with 5 nodes, corresponding to 5 sleep stages to be classified. The last dense layer utilizes softmax activation function in the output layer to generate probability prediction for the 5 sleeping stages.

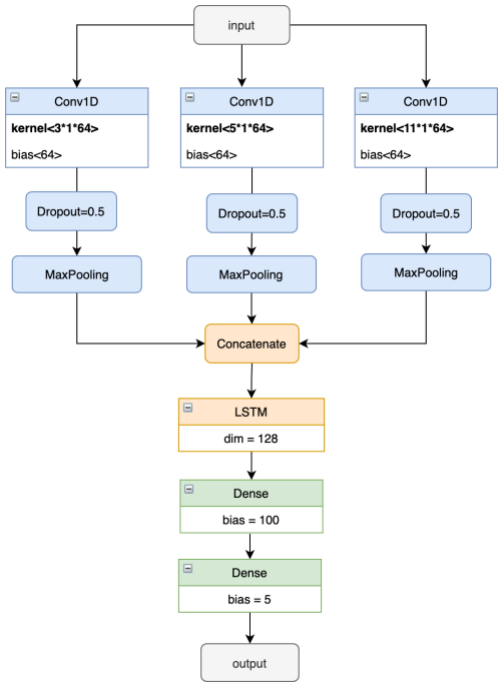

Fig. 1. The optimized design of a hybrid CNN-LSTM architecture

IV. Implementation

The proposed DL approach is demonstrated in three sub experiments with optimizations on top of the previous one. project will implement 3 DL models step by step. Specifically, I will investigate the following questions

A. *Data Preprocessing*

As mentioned earlier, single-channel EEG which has been widely utilized for sleep staging is applied in this project. A collection of 20 healthy individual records, including 10 women subjects and 10 men subjects, each with two nights of full sleep were prepared and analysed. In the extracted dataset, 30 min of wake stage data were retained from before the first sleep epoch and from after the final sleep epoch for every record. Besides, stage N3 and N4 were merged into a single slow-wave stage. The sleep stages are classified according to epoch times of 30s. Since the sampling rate is 100HZ, there are 3000 points of EEG data in each fragment.

Basically, the raw data was firstly transformed into parquet format, which is a column-based data format designed for parallel processing. Then Python DataFrame was used to read raw data from parquet data followed by data filtering and transformation. Finally, data was split into training set, validating set and testing set at a proportion of 80:10:10. This time series data is then fed into models implemented in 3 experiments.

B. *Model Implementation*

Inspired by the speech emotion recognition project [7] publicly available on Github https://github.com/vandana-rajan/1D-Speech-Emotion-Recognition/, three experiments based on CNN and LSTM are conducted in my project. Instead of using only 1 head, multi-head mechanism, similar to multi-head attention mechanism in Transformer [6], is utilized to

capture intrinsic signal characteristics with various time scales. Also, dropout is added in my experiments to mitigate the overfitting problem.

It is worth noting that 4 callbacks are added in this project as shown in Table I.

TABLE I. TABLE TYPE STYLES

Callbacks	*Use*
tensorboard_callback	Write Tensorboard logs after every epoch of training to monitor the metrics.
checkpoint	Save the model that has the best performance observed during training for later use.
early_stopping	Halt the training process at the right time once triggered.
reduce_lr_on_plateau	Monitor the specified metric and adjust the learning rate when necessary

Callbacks are used to control the training procedure to retrieve the internal states and statistics during model training.

- ***tensorboard_callback****:* Generate logs for tensorboard which allows visualizaing dynamic graphs of the training and testing metrics, as well as activation histograms for the different layers in the defined model. In my experiments, those logs are saved to the logs file under the code folder.
- ***checkpoint****:* Periodically save the model that has the best performance to disk. In my experiments, the metric used for this callback is sparse_categorical_accuracy and the checkpoint model is saved to the out/saved_models under the code folder.
- ***early_stopping****:* Too many epochs can lead to overfitting of the training dataset. Early stopping is of great use to solve the overfitting problem as an effort to halt the training process once the model performance stops improving on a patience threshold. In my experiments, the training process will stop if the model does not show significant improvent in performance inside 20 epochs.
- ***reduce_lr_on_plateau****:* Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is witnessed for a 'patience' number of epochs, the learning rate is reduced accordingly. In my experient, the patience is set to 20 and the metric of sparse_categorical_accuracy will be monitored as the labels are encoded into in one-hot format. the quantity is set to 20.

The code tree structure of this project is shown in Fig. 2 below. All the concrete model definitions is organized under the models folder. Model cnn1head, cnn3head and cnn3head_lstm all inherit base_model where all the necessary facilities like callbacks and fit_model functions are all abstracted. The script run.py is used to revoke those models defined in the aforementioned models package.

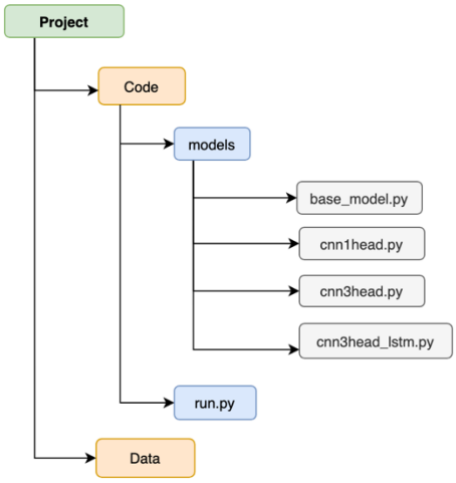

Fig. 2. Code tree structure for the project

In the following part, the models implemented will be elaborated. The model diagrams are drawn according to the model invocations during the running process with plot_model in keras.

1) ***Model CNN1HEAD***

Initially, a simple CNN model which only consists of 1D convolutional layer with 64 filters in a size of 5 was implemented to probe its ability in sleep staging and build the learning pipeline.

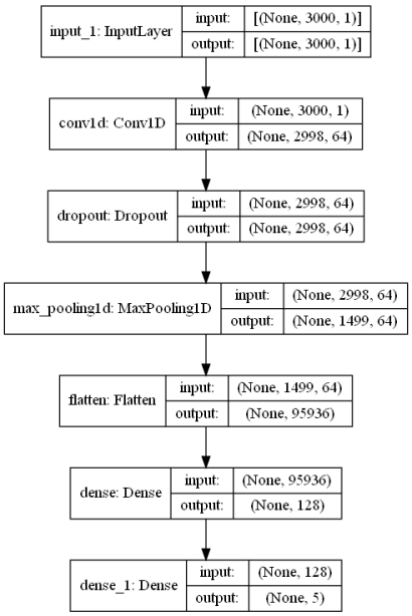

Fig. 3. CNN1HEAD model architecture

As shown in Fig. 3, there is a Dropout and 1D MaxPooling layers following the convolutional layer. The model is then connected to a dense layer with 5 nodes corresponding to 5 sleep stages to be identified. The last dense layer utilizes softmax activation function to generate probability

prediction for the 5 sleeping stages. The model architecture generated during training is show in Fig. 3 below. However, this model only achieves an accuracy of 51.82%.

*2) **Model CNN3HEAD***

Encouraged by the preliminary result, a different CNN structure that combines multiple smaller CNN models with different kernel sizes was then probed. This multi-head mechanism, similar to multi-head attention mechanism in Transformer [6], is leveraged to capture intrinsic signal characteristics with various time scales. The other part is similar to the original architecture as described in Model CNN1HEAD. Fig. 4 shows this updated version of CNN model with 3 heads. Favourably, this multi-head CNN model yields better results than the baseline CNN model with an accuracy of 73.9%.

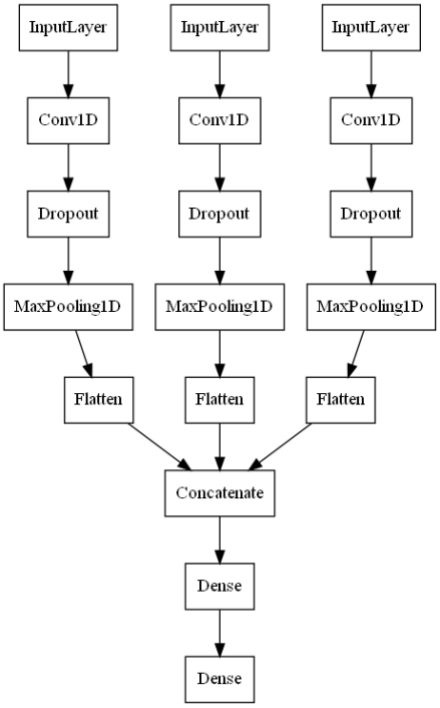

Fig. 4. CNN3HEAD model architecture

*3) **Model CNN3HEAD_LSTM***

This is the optimized version of model based on the suggestions from our TA - Dr. Qiao in the code demo. The code change afterwards is that I added another LSTM layer following the 3-head CNN layer considering that LSTM is good at tackling sequence. The other layers stay the same as that of model CNN3HEAD. The architecture of this newest model is shown in Fig. 5. This further optimized model outperforms the previous two CNN models mentioned above by 82.143% accuracy.

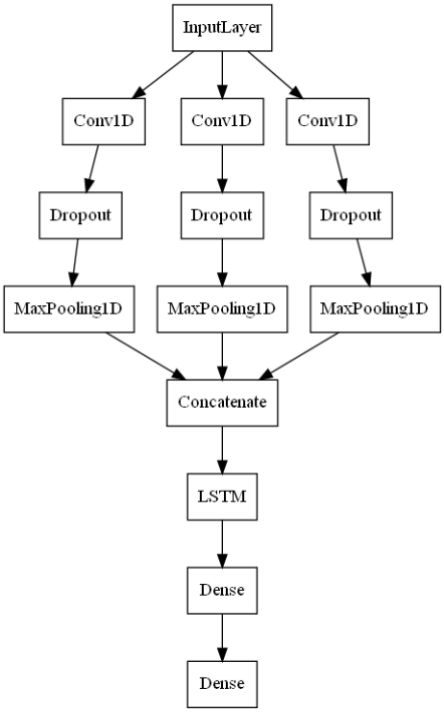

Fig. 5. CNN1HEAD model architecture

C. *Experimental Environment*

The model is developed in Python 3.7 on PyCharm using TensorFlow Keras package. In the early exploration/setup stage, the developed model was tested with a smaller dataset consisting only 7 data files and a few epochs as shown in the project code demo to get myself familiar with the data and establish the basic structure of the data processing and learning pipeline.

In the later stage of this project, the proposed method was run against the whole dataset with 8G data and 55 epochs with Tensorflow-GPU on the Cedar Cluster supported by ComputeCanada, which is a platform facilitating the research and innovation by making advanced research computing resources accessible.

D. *Results*

In all my experiments, the initial learning rate is 0.001, which will decrease if it does not see a significant improvement in accuracy for 20 epochs. 55 epochs were set for all these 3 models and the dropout rate is 0.5.

Table II shows the comparison of the three implemented models. The hybrid architecture of CNN and LSTM does obtain a satisfactory accuracy of 82.143%.

TABLE II. PEFORMANCE COMPARISON

Models	***Learning rate***	***Epochs***	***Batch size***	***Accuracy***
cnn1head	initial 0.001	55	16	0.518
cnn3head	initial 0.001	55	16	0.739
cnn3head_lstm	initial 0.001	55	16	0.821

Starting from model CNN1HEAD, only 51.82 of accuracy obtained. After applying 3 heads to the original mode, the accuracy soars up to 73.9% as shown in Fig. 6.

```
Epoch 00019: sparse_categorical_accuracy improved from 0.72752 to 0.73286
Epoch 20/20
7791/7791 - 49s - loss: 0.6493 - sparse_categorical_accuracy: 0.7390
```

Fig. 6. CNN3HEAD accuracy

With an additional LSTM layer stacked after the concatenation of the 3-head CNN, the accuracy further jumps from 73.9% to 82.14% as shown in Fig. 7. This indicates that the LSTM plays an essential role in improving the model's accuracy in classifying the sleep stage.

```
Epoch 00047: sparse_categorical_accuracy did not improve from 0.82143

Epoch 00047: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
Epoch 00047: early stopping
```

Fig. 7. CNN3HEAD_LSTM accuracy

Next, Tensorboard is utilized to visualize the losses and sparse categorical accuracy based on the logs saved during the validation.

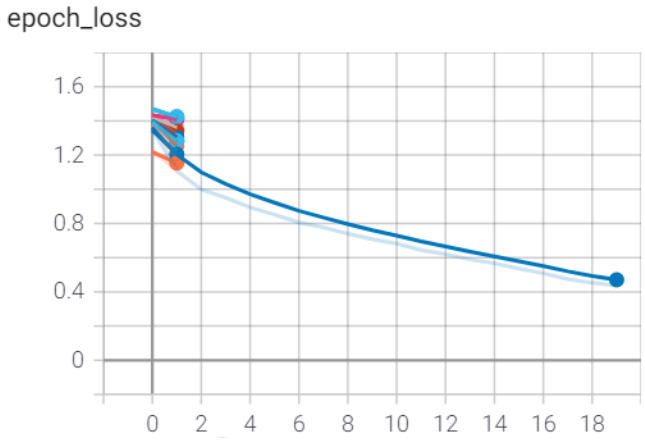

Fig. 8. CNN3HEAD losses

It shows in Fig. 8 that the loss decreases gradually along with the changes of epochs.

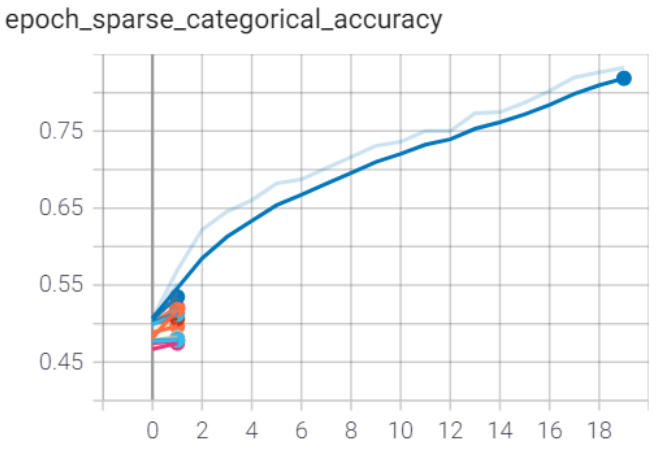

Fig. 9. CNN3HEAD accuracy

It is obsearved form Fig. 9 that the accuracy of CNN3HEAD increases along with the changes of epochs.

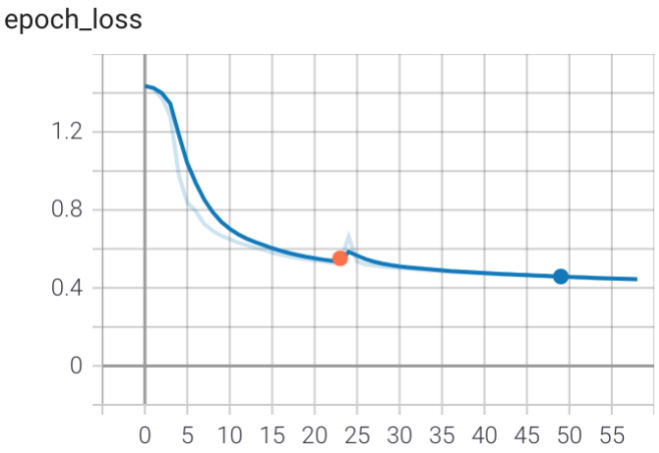

Fig. 10. CNN3HEAD_LSTM losses

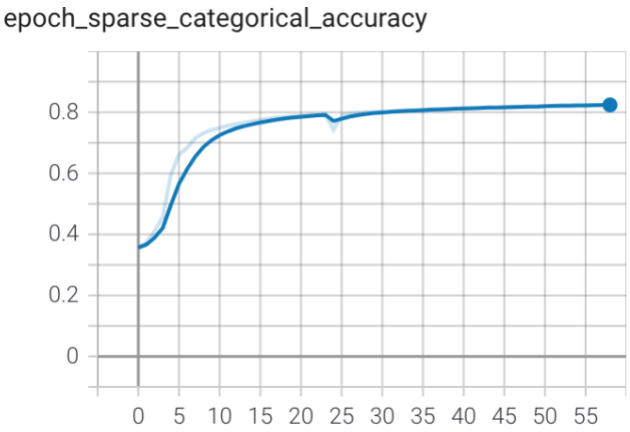

Fig. 11. CNN3HEAD_LSTM accuracy

Both Fig. 10 and Fig. 11 witness a steady point around 25 epochs. Loss and accuracy did not change much after this point. However, they were updated much faster than that of Fig. 8 and Fig. 9 in the beginning. Therefore, the model has benefited from LSTM layer in terms of the performance and efficiency.

E. Critical Discussion

The biggest challenge lies in this project is the size of data, albeit only single-channel EEG data is applied in my project. Therefore, I set up the learning pipeline with only a small portion of the data and then dump all the data to the cedar cluster when the model is running on Tensorflow-GPU.

I have attempted to improve the accuracy by tuning the hyper-parameters. However, no significant improvement is seen. The conjecture is that CNN might perform well on image processing, but it is not suitable for handling time series data as shown by the results from model 1 and model 2. However, the combination of different neural networks is capable of improving the model performance.

Regarding the number of parallel heads, I set it in the experiments to 3 which can be always set much bigger if environment permits.

With much lower number of trainable parameters and fewer layers and neurons based on purely EEG data, this simple hybrid model still behaves moderately well with 82.143% accuracy compared to the state-of-the-art sleep stage classifiers.

This project has many implications. The sleep staging process can be automated in a much more efficient and economical way. If we make good use of CNN combined with RNN and run it againest much larger datasets of thousands of subject recordings, workload that doctors and specialists have been bearing for many years can be reduced sharply.

V. Conclusion

This project introduced a deep learning technique based on the combined architecture of multi-head CNN and LSTM to identify sleep stages from single-channel EEG signal. Specifically, multi-head CNN is used for deep feature extraction followed by a LSTM layer used for prediction. The experimental results show that this system achieves an accuracy of 73.9% for pure 3-head CNN and 82.143% for the mixed approach of 3-head CNN and LSTM. The conjecture is that CNN might perform well on image processing, but it is not suitable for handling time series data as shown by the results from model 2 in this project. Experiments also verified that the hybrid model has much lower number of trainable parameters and fewer layers while still performs moderately well, which fully satisfied the objective of this project.

Future work will be focused on utilizing unsupervised learning algorithm - autoencoder - as a compensatory strategy to the current approach since labeled data is difficult to collect. In order to evaluate the general performance of the proposed hybrid approach for sleep staging in a realistic setting, the model is awaiting to be tested on external clinical data, and transfer learning is to be applied to improve the generalization of the proposed hybrid model. Sensitivity analysis based on number of CNN heads is also another direction in the future.

Acknowledgment

The author thanks Prof. Zulkernine and Dr. Qiao for their discussion and guidance.

References

[1] H. Korkalainen *et al*., "Accurate Deep Learning-Based Sleep Staging in a Clinical Population With Suspected Obstructive Sleep Apnea," in *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 7, pp. 2073-2081, July 2020, doi: 10.1109/JBHI.2019.2951346.

[2] E. Alickovic and A. Subasi, "Ensemble SVM Method for Automatic Sleep Stage Classification," in *IEEE Transactions on Instrumentation and Measurement*, vol. 67, no. 6, pp. 1258-1265, June 2018, doi: 10.1109/TIM.2018.2799059.

[3] G. Zhu, Y. Li and P. Wen, "Analysis and Classification of Sleep Stages Based on Difference Visibility Graphs From a Single-Channel EEG Signal," in *IEEE Journal of* Biomedical and Health Informatics, vol. 18, no. 6, pp. 1813-1821, Nov. 2014, doi: 10.1109/JBHI.2014.2303991.

[4] I. N. Yulita, M. I. Fanany and A. M. Arymurthy, "Combining deep belief networks and bidirectional long short-term memory: Case study: Sleep stage classification," *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, Yogyakarta, 2017, pp. 1-6, doi: 10.1109/EECSI.2017.8239089.

[5] B. Kemp. (2013, Jun.). The Sleep-EDF Database [Onlne]. Available: http://www.physionet.org/physiobank/database/sleep-edf/

[6] Vaswani, Ashish & Shazeer, Noam & Parmar, Niki & Uszkoreit, Jakob & Jones, Llion & Gomez, Aidan & Kaiser, Lukasz & Polosukhin, Illia. (2017). Attention Is All You Need.

[7] https://github.com/vandana-rajan/1D-Speech-Emotion-Recognition/.