# Deep Learning for Sleep Stage Classification

Yang Wang, Dongrui Wu
Key Laboratory of the Ministry of Education for Image Processing and Intelligent Control,
School of Automation, Huazhong University of Science and Technology, Wuhan, China
Email: wangyang_sky@hust.edu.cn, drwu@hust.edu.cn

***Abstract*—Scoring of sleep stages plays an important role in the diagnosis of sleep-related diseases. Scoring by visual inspection is time-consuming and heavily depends on the experience of experts. Thus, there is an urgent need for an automatic sleep stage classification system. This paper proposes a novel compact convolutional neural network (C-CNN) using only single-channel EEG signal. Compared with traditional machine learning approaches based on hand-engineered features, our approach provides an end-to-end solution that requires almost no prior knowledge and preprocessing while achieving better performance. Experiments on the expanded Sleep-EDF database verified its effectiveness and efficiency. In addition, we notice the issue of class imbalance in sleep stages, and a class-imbalance metric, the balanced classification accuracy (BCA), is introduced. At the cost of a little drop in accuracy, which is still higher than existing classification methods, the introduction of class-imbalance weights can significantly increase the BCA metric and result in a higher recall for each sleep stage. This paper also proposes a recurrent neural network based on the attention mechanism and bidirectional long short-term memory (LSTM), which provides better performance than C-CNN but requires more training time.**

***Index Terms*—Sleep stage classification, convolutional neural network, attention, long short-term memory, class imbalance**

## I. INTRODUCTION

Almost one-third of a person's life is spent on sleep. During sleep, most of the body's systems are in an anabolic state that helps restore immune, nervous, skeletal and muscular systems. Thus, sleep plays an important role in human health.

However, lots of people have sleep problems, e.g., at least 10% of the population in America suffer from sleep disorders [16]. Proper scoring of sleep stages can help diagnose sleep disorders and track the effect of treatment [4]. Polysomnography (PSG) is the gold standard for sleep quality assessment. It usually requires the subject to wear multiple sensors and record various physiological signals. The PSG recording is then segmented into 30s epochs, which are visually inspected by a sleep expert and sorted according to protocols such as the traditional Rechtschaffen and Kales (R&K) criteria [17] and those proposed by the American Academy of Sleep Medicine (AASM) [2]. This manual scoring process is time-consuming, and the results may be inconsistent among different experts. A study shows that the average agreement rate of sleep stage scoring among experts is only 82.6% [18]. Thus, automatic sleep state classification is desired.

There are already multiple machine learning approaches for automatic sleep stage scoring using hand-engineered features [3], [8], which usually performs preprocessing to remove artifacts and noise, feature extraction and selection to obtain discriminative features, and finally machine learning to train a classifier. The scoring performances of these approaches heavily depend on the quality of the hand-engineered features, which are limited by the designer's experience, and can hardly be optimal.

In contrast to the traditional feature engineering plus machine learning approaches, deep learning provides an end-to-end solution that can automatically mine the relationship between the raw input and the output. It has achieved great success in a wide range of applications, including image processing [20], video analysis [24], natural language processing [1], [15], and so on. Not surprisingly, multiple deep learning approaches, e.g., autoencoders [22], convolutional neural networks (CNNs) [5], and recurrent neural networks (RNNs) [21], have also been proposed for sleep stage classification.

Additionally, different sleep stages do not occur with equal probability. If no special attention is paid to the class imbalance problem, then the minority stages may be ignored. Most deep learning based sleep stage classification approaches attempted to address this issue by resampling the data (oversampling the minority classes or undersampling the majority classes) so that different classes are balanced [5], [21], [22]. However, resampling data modifies the distribution of the raw data and may cause other problems [10]. For example, oversampling increases the total number of samples, and hence increases the computational cost; undersampling removes certain samples, and hence loses information. In this paper we cope with class imbalance problem using cost-sensitive learning [7], i.e., we assign larger weights to the minority classes so that they will not be overwhelmed.

In summary, the main contributions of this paper are:

1) We propose two novel deep learning models for sleep stage classification. The first, based on CNN, is compact and efficient. The second, based on LSTM, can achieve better performance at the cost of longer training time.
2) We integrate cost-sensitive learning with deep learning to handle class-imbalance in sleep stage classification, and achieve better balanced classification accuracy (BCA).

The remainder of this paper is organized as follows: Section II introduces the sleep dataset and proposes our approaches. Section III introduces the experiment settings and compares the performances of our proposed models with several state-of-the-art approaches. Section IV draws conclusions.

## II. Methods

### A. Dataset

The SC (Sleep Cassette) files from the expanded Sleep-EDF database [14] on PhysioNet [9] were used in our study. They are PSG recordings of 20 healthy subjects (10 males and 10 females, 25-34 years old) without any sleep-related diseases. 19 subjects had PSG recordings for two consecutive days and one subject only had one day's recording. Each recording lasted for 20 hours. For consistency, we only used data from the 19 subjects with two days' recording.

Each PSG recording contained EEG (from Fpz-Cz and Pz-Cz pairs), horizontal EOG, submental chin EMG, and oro-nasal respiration. The sampling frequency of EEG and MEG was 100Hz. Each 30-s epoch was manually classified into one of the following classes according to the R&M criteria: movement time (M), wakefulness (W), rapid eye movement (REM), non-REM including Stage 1 (S1), Stage 2 (S2), Stage 3 (S4) and Stage 4 (S4), and not scored. This paper considered only W, REM, and S1-S4, and only used the EEG signal from the Fpz-Cz pair[1]. The number of available epochs for each sleep stage is shown in Table I.

TABLE I
NUMBER AND PERCENTAGE OF EEG EPOCHS IN EACH SLEEP STAGE.

| | W | S1 | S2 | S3 | S4 | REM | Total |
|---|---|---|---|---|---|---|---|
| Number | 70,450 | 2,731 | 17,302 | 3,307 | 2,249 | 7,545 | 103,600 |
| Percentage | 68.00 | 2.65 | 16.70 | 3.19 | 2.17 | 7.28 | 100.00 |

### B. Flowchart

The flowchart of our proposed approaches is shown in Fig 1. For each 30-s EEG epoch, we first perform preprocessing to transform it into a power spectral density map $\boldsymbol{X}$, which is then fed into the Feature Extractor. Different configurations of the Feature Extractor are introduced below, and dropout [12] was used after all such extractors to reduce overfitting. Then, a fully-connected layer with 128 units (FC-128) is used to combine these features, and a softmax layer gives the final prediction.

### C. Preprocessing

We performed some preprocessing on the raw EEG signal.

Given a 30-s epoch, we first converted it into a power spectral density (PSD) map using short-time Fourier transform (STFT) with a moving Hamming window of size 1s and overlap of 0.5s. Then we converted the powers to dBs. Thus each 30-s epoch was converted to a PSD map $\boldsymbol{X} \in R^{T\times F}$, where $T$ (time) was 59 and $F$ (frequency) was 51. An example of the EEG signal after conversion is shown in Fig. 2. It should be noted that since the sampling frequency of the EEG signal was 100 Hz, according to the Nyquist sampling theory, the maximum frequency of the recoverable signal is 50 Hz.

[1]We also performed experiments on the Pz-Oz pair. The results were slightly worse.

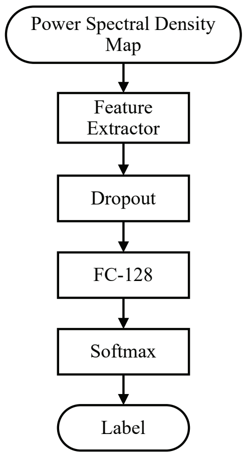

Fig. 1. The flowchart of our proposed approaches.

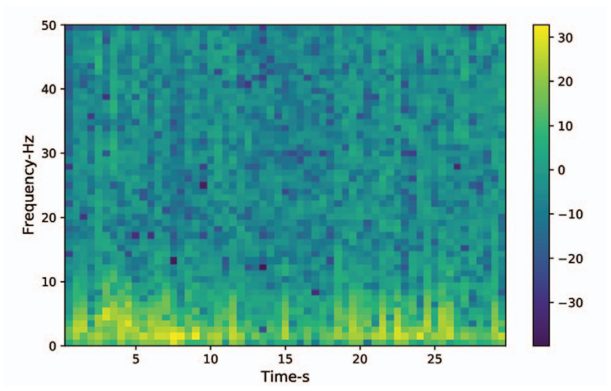

Fig. 2. Power spectral density map.

Next, $z$-normalization was performed to make each dimension of the feature to have mean zero and standard deviation one.

### D. Baseline Feature Extractor

Our baseline feature extractor is similar to CNN-static (CNN-s) in [15]. This model was originally proposed for natural language processing, and has been widely adopted in other areas for its relatively simple structure and competitive performance.

The main part of the model is shown in Fig. 3. For natural language processing, a pre-trained word vector of a specific length is used to represent each word. Words in the same sentence are replaced with the corresponding word vector in order. The resulting word vector sequence is used as the model's input, which is quite similar to the PSD map we got: The word vectors in the sequence correspond to the PSD vectors in our map, and the order of word vectors corresponds to the time sequence of PSD vectors.

Therefore, we adopted the CNN-s from [15] as the baseline feature extractor.

$Conv1D(3\times F-128)$ in Fig. 3 represents a 1D convolutional layer with 128 filters of size $3\times F$, where $F$ is the length

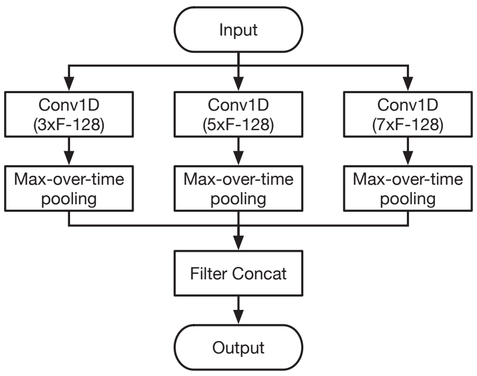

Fig. 3. The baseline feature extractor (A-CNN).

of the PSD vector. The convolution is also called temporal convolution because the filter is moving in the time dimension of the PSD map in the convolution.

Different filter sizes could be used to obtain receptive fields of different sizes. We tried several different sizes and obtained similar results. Filter sizes 3, 5 and 7 were used in our experiments. The filter stride was 1 and the activation function was rectified linear unit (ReLU).

For the feature map obtained after convolution, a max-over-time pooling operation with stride 1 was applied, i.e., selecting the largest value in the time dimension as the feature corresponding to a particular filter. The idea here was to capture the most important feature (maximum value) for each filter. Finally, features of different scales were concatenated together for further processing.

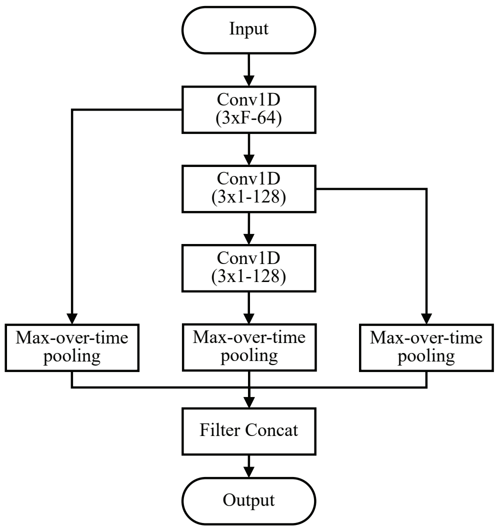

Fig. 4. The proposed C-CNN feature extractor.

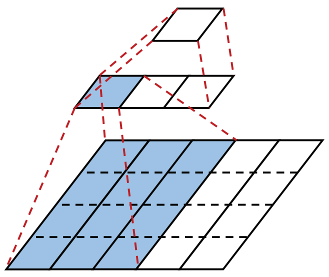

Fig. 5. Stacking a $3 \times F$ convolutional layer and a $3 \times 1$ layer is equivalent to a $5 \times F$ layer.

### *E. Feature Extractor 2*

*1) CNN Feature Extractor:* We propose a new feature extractor, as shown in Fig. 4, to enhance CNN-s.

Instead of using large receptive fields (or filters) of size $5 \times F$ and $7 \times F$, we only use receptive fields of size $3 \times F$ and $3 \times 1$. This idea is adopted from [20]. Stacking a $3 \times F$ convolutional layer and a $3 \times 1$ layer achieves an effective receptive field of $5 \times F$, as illustrated in Fig. 5. Similarly, stacking a $3 \times F$ layer and two $3 \times 1$ layers results in a $7 \times F$ effective receptive field.

Stacking three convolutional layers instead of using a single $7 \times F$ layer directly offers at least two benefits: first, three ReLU layers instead of one are now used, resulting in more nonlinearity; second, fewer parameters are needed, making the network more compact. Assume the network input and output channels of the convolution layer are both $C$, then the number of parameters required for a separate $7 \times F$ convolution layer is $7 \times F \times C^2$, while the stacked convolution layers is $3 \times (F + 1 + 1) \times C^2$, i.e. 43% of the former.

Moreover, in order to utilize features of different levels, we used the feature map from middle layers for the final prediction. We thought that pushing useful gradients to the lower layers, making them immediately useful, can make the training more stable and converge faster. Batch normalization was adopted after each convolutional layer, which is proved to be useful in handling internal covariate shift and making the subsequent training easier. So we got the final feature extractor shown in Fig. 4.

*2) Attention Feature Extractor:* Except for CNN, we also applied LSTM [13] with attention mechanism, which has been widely used in time series and natural language processing [1], to sleep stage classification and proposed the Attention feature extractor in Fig. 6.

RNN has a strong ability to capture temporal information, and EEG data is a time series. Therefore, RNN is naturally suitable for such a scenario. Given the input sequence $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{T'})$, the RNN calculates the hidden state sequence $\boldsymbol{H} = (\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_{T'})$ by iterating the following equation from $t = 1$ to $T^{'}$:

$$\boldsymbol{h}_t = f(\boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{b}) \tag{1}$$

where $\boldsymbol{W}_x$ and $\boldsymbol{W}_h$ are weight matrixes, $\boldsymbol{b}$ is a bias vector, $f$ is an activation function, and all of them are shared at different time steps. Thus, the current hidden layer state $\boldsymbol{h}_t$ of RNN is not only related to the current input $\boldsymbol{x}_t$, but also related to the previous hidden layer state $\boldsymbol{h}_{t-1}$. That's why RNN can capture the time dependency.

LSTM is a special RNN, which was introduced by Hochreiter & Schmidhuber at 1997. It uses multiple gates to selec-

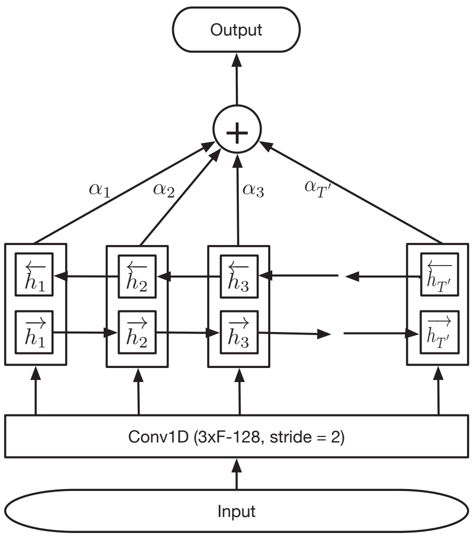

Fig. 6. Attention feature extractor.

tively store memory, thus it is more efficient at capturing long-term dependencies. In this paper, we further used the bidirectional LSTM with attention mechanism. To the best of our knowledge, it has not been used in sleep stage classification.

When training a regular LSTM, there is a limitation that future input information cannot be used. However, bidirectional LSTM [19] could solve this issue. It consists of forward and backward LSTMs, which are trained simultaneously in the positive and negative time direction, respectively. By concatenating the forward hidden state $\overrightarrow{\boldsymbol{h}_t}$ and the backward one $\overleftarrow{\boldsymbol{h}}_t$, which contains future input information, we obtain $\boldsymbol{h}_t$ as the hidden state of the bidirectional LSTM, i.e., $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}_t}; \overleftarrow{\boldsymbol{h}_t}]$. 128 LSTM units were used.

As for the attention mechanism, it has been widely adopted in various of applications, such as machine translation [1], image caption generation [23], question answer [11], etc. When combined with LSTM, its output, context vector $\boldsymbol{c}$, is computed as a weighted sum of each hidden state $\boldsymbol{h}_t$ across different time steps:

$$\boldsymbol{c} = \sum_{t=1}^{T'} \alpha_t \boldsymbol{h}_t \tag{2}$$

The weight $\alpha_t$ of each $\boldsymbol{h}_t$ is computed by

$$\alpha_t = \frac{exp(e_t)}{\sum_{t=1}^{T'} exp(e_t)} \tag{3}$$

where $e_t$ can be given by a trainable fully-connected layer with $\boldsymbol{h}_t$ as input.

Besides, in order to avoid the problem of exploding or vanishing gradients during the training of LSTM, we firstly did a 1D convolution with stride 2 in the input to reduce the sequence length, and then adopted batch normalization. Thus, we got the final Attention feature extractor in Fig. 6.

### F. Network Training

The training goal of our networks was to minimize the cross-entropy loss function. We trained these models using an Adam optimizer with a learning rate $10^{-3}$. And a decay rate of 0.95 was adopted to the learning rate per pass over the training set. The batch size was set to 128. Dropout is an effective regularization method for reducing overfitting, so we applied dropout with drop rate 0.5 before the fully-connected layer. In addition, early stopping was used to determine when to stop training, by monitoring the model's accuracy on a randomly selected validation set. Our models were implemented using TensorFlow and trained with a single Nvidia GeForce GTX 1080 GPU.

### G. The Class-imbalance Weight

When training a model with imbalanced data, what typically happens is that the learned model tends to predict samples to majority class. So we adopted cost-sensitive learning and added a class-imbalance weight in the loss function to reweight the cost of making a wrong prediction. The general principle is assigning larger weights to the minority classes. For each class, the weight is computed as,

$$w_l = \frac{max\{N_l\}_{l=1}^{L}}{N_l} \tag{4}$$

where $N_l$ is the number of samples in Class $l$, and $L$ is the number of classes.

The motivation for this is that we expect the model to pay equal attention to the class with fewer samples (the class with more samples would be seen more frequently by model during training), and ensure that each class has a similar prediction accuracy. Except for the R&M criteria, we also merged S3 and S4 according to the AASM manual. Moreover, we also considered a simpler task that S1 and S2 were merged.

## III. Experiments

### A. Performance Measures

To evaluate the performance of the proposed models, we used recall (RE), which is also known as sensitivity, overall accuracy (ACC) and Cohen's Kappa coefficient (Kappa) [6] as performance measures. Considering serious class imbalance in the dataset, we introduced a class-imbalance metric, i.e. the balanced classification accuracy (BCA). The RE, ACC and BCA are defined as follows:

$$RE_l = \frac{TP_l}{N_l} \tag{5}$$

$$ACC = \frac{\sum_{l=1}^{L} TP_l}{N} \tag{6}$$

$$BCA = \frac{1}{L}\sum_{l=1}^{L} \frac{TP_l}{N_l} = \frac{\sum_{l=1}^{L} RE_l}{L} \tag{7}$$

where $TP_l$ represents true positives for Class $l$, $L$ is the number of classes, $N$ is the number of all samples, and $N_l$ represents the number of samples in Class $l$.

In fact, the $RE_l$ represents the proportion of samples in Class $l$ that are predicted correctly, while ACC represents the proportion of samples across all classes that are predicted correctly. We expect each class to have a similar prediction accuracy (i.e. recall), so we used the BCA, which is the mean of recalls for different classes.

Besides, since there might exist obvious differences between the EEG signal of different subjects, we performed the leave-one-subject-out cross validation.

## B. Results for Different CNN Feature Extractors

In this section, the performances of different CNN feature extractors for six classification task were compared. The average ACCs and BCAs are shown in Table II.

TABLE II
AVERAGE PERFORMANCE OF DIFFERENT CNN FEATURE EXTRACTORS ACROSS THE 19 SUBJECTS.

| Feature Extractor | ACC | BCA |
|---|---|---|
| A-CNN | 87.37 | 68.80 |
| C-CNN | 88.19 | 70.95 |

As seen from the table, the baseline A-CNN already achieved high classification accuracy. The performance of C-CNN formed by the stacking of convolutional layers with small receptive fields was further improved, which was mainly benefited from its deeper layers and stronger nonlinear fitting capability.

## C. The Impact of Class-imbalance Weight

In this section, we explored the impact of class-imbalance weight on performance metrics. It was a six class classification task, using the C-CNN feature extractor. The only difference between the two experiments was whether the class-imbalance weight was used in the loss function. The result is shown in Table III. $Mean$ means the performance is the average of different subjects, while $Summary$ is calculated after combining all the predictions of different subjects.

TABLE III
THE IMPACT OF CLASS-IMBALANCE WEIGHT

| 6-class | Mean | | Summary | | |
|---|---|---|---|---|---|
| | ACC | BCA | ACC | BCA | Kappa |
| Weighted | 88.19 | 70.95 | 88.24 | 73.21 | 77.44 |
| Not weighted | 90.94 | 67.94 | 90.98 | 70.18 | 81.97 |

We can observe that the class-imbalance weights had a great influence on ACC and BCA. After applying the weights, the BCA metric increased significantly, while the ACC and Kappa decreased, which was mainly due to the decreased prediction accuracy for those categories with more samples. When the classes are significantly unbalanced, sometimes we would like all classes to have similar recalls (i.e., higher BCA), even if some classes only have a small number of samples.

## D. Comparison between C-CNN and Attention Feature Extractors

Convolutional neural network is good at processing image information, while recurrent neural network is better for time series task.

We further compared the performance of C-CNN and Attention feature extractor under multiple classification tasks. The results are shown in Table IV.

TABLE IV
COMPARISON BETWEEN C-CNN AND ATTENTION FEATURE EXTRACTORS.

| | 6-class | | 5-class | | 4-class | |
|---|---|---|---|---|---|---|
| Model | ACC | BCA | ACC | BCA | ACC | BCA |
| C-CNN | 88.19 | 70.95 | 90.23 | 77.09 | 91.03 | 85.94 |
| Attention | 88.43 | 73.58 | 90.27 | 79.50 | 91.91 | 87.56 |

It shows that the performance of the Attention feature extractor was always a little better than C-CNN, both on ACC and BCA, which means it learned better representations. However, due to the characteristics of LSTM itself, the training time Attention needed was much more than C-CNN.

## E. Comparison with Exiting Methods

We also compared our approaches with two exiting methods and demonstrated that our approaches outperformed them. The results are shown in the Table V.

The review by Boostani et al. [3] compared five methods based on hand-engineered features using the expanded Sleep-EDF database, and [8] which used entropy of continuous wavelet transform as features and random forest as classifier achieved the best performance. It got 87.06% ACC and 71.10% BCA respectively (the BCA was computed based on the confusion matrix in the paper).

Compared with the traditional methods based on hand-engineered features, our proposed C-CNN provided an end-to-end solution that requires almost no prior knowledge, and achieved improvement of +3.17% (from 87.06% to 90.23%) and +6.12% (from 71.10% to 77.22%) in ACC and BCA, respectively. Besides, our approach could further improve ACC (91.88%) and still provide a compare BCA (73.83%) if the class-imbalance weight was not adopted.

DeepSleppNet [21] was the state-of-the-art model for sleep stage scoring. This model was divided into two parts. The first part was representation learning using CNN, and the second part used adjacent EEG epoches and bidirectional LSTM to learn transition rules among sleep stages. The training of DeepSleepNet was also divided into two steps. The first step performed a supervised pre-training on the representation learning part using a class-balanced set generated by resampling; The second steps performed fine-tuning on the whole model using a sequential training set, which meant it used adjacent EEG epochs to train the bidirectional LSTM part.

We re-implemented the first part for comparison in our experiments. Compared with DeepSleppNet, our models were much simpler, required no special handing and achieved a slightly higher ACC while providing a comparable BCA.

TABLE V
COMPARISON WITH THE EXITING METHODS

| | | 6-class | | | | | 5-class | | | | | 4-class | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Mean[1] | | Summary[2] | | | Mean[1] | | Summary[2] | | | Mean[1] | | Summary[2] | | |
| | | ACC | BCA | ACC | BCA | Kappa | ACC | BCA | ACC | BCA | Kappa | ACC | BCA | ACC | BCA | Kappa |
| Our proposed C-CNN | Weighted | 88.19 | 70.95 | 88.24 | 73.21 | 77.44 | 90.23 | 77.09 | 90.26 | 77.22 | 81.07 | 91.03 | 85.94 | 91.04 | 85.68 | 82.22 |
| | No Weighted | 90.94 | 67.94 | 90.98 | 70.18 | 81.97 | 91.84 | 73.67 | 91.88 | 73.83 | 83.71 | 91.84 | 73.67 | 92.43 | 84.63 | 84.70 |
| Our proposed Attention | Weighted | 88.43 | 73.58 | 88.47 | 75.89 | 77.99 | 90.27 | 79.50 | 90.30 | 79.77 | 81.26 | 91.91 | 87.56 | 91.93 | 87.63 | 83.92 |
| Review [3] | | | | | | | 87.06 | | | 71.10 | | | | | | |
| DeepSleepNet [21] | | | | | | | 90.01 | 79.39 | 90.04 | 79.75 | 80.86 | | | | | |

[1] The results of $Mean$ were the average of different subjects.
[2] The results of $Summary$ were computed after combining all the predictions of different subjects.

Besides, the training time C-CNN required was only half of DeepSleepNet.

## IV. CONCLUSIONS

This paper has proposed two novel sleep state classification approaches based on single-channel EEG signal: C-CNN using small receptive fields and multi-level features, and Attention method based on the attention mechanism and bidirectional LSTM. The latter outperformed the former at the cost of longer training time. We also introduced the BCA metric and integrated cost-sensitive learning with deep learning to handle class-imbalance in sleep stage classification. Experiments verified that our proposed models are efficient, and they outperformed an existing approach using hand-engineered features, and also a state-of-the-art deep learning model.

## ACKNOWLEDGEMENT

This research was supported by the National Natural Science Foundation of China (61873321).

## REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014. [Online]. Available: http://arxiv.org/abs/1409.0473

[2] R. B. Berry, R. Brooks, C. E. Gamaldo, S. M. Harding, C. Marcus, B. Vaughn *et al.*, "The AASM manual for the scoring of sleep and associated events," *Rules, Terminology and Technical Specifications, Darien, Illinois, American Academy of Sleep Medicine*, 2012.

[3] R. Boostani, F. Karimzadeh, and M. Nami, "A comparative review on sleep stage classification methods in patients and healthy individuals," *Computer Methods and Programs in Biomedicine*, vol. 140, pp. 77–91, 2017.

[4] M. A. Carskadon and A. Rechtschaffen, "Monitoring and staging human sleep," *Principles and Practice of Sleep Medicine*, vol. 3, pp. 1197–1215, 2000.

[5] S. Chambon, M. Galtier, P. Arnal, G. Wainrib, and A. Gramfort, "A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series." *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, no. 4, pp. 758–769, 2018.

[6] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[7] C. Elkan, "The foundations of cost-sensitive learning," in *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence IJCAI*, Seattle, WA, August 2001, pp. 973–978.

[9] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[8] L. Fraiwan, K. Lweesy, N. Khasawneh, H. Wenz, and H. Dickhaus, "Automated sleep stage identification system based on time–frequency analysis of a single eeg channel and random forest classifier," *Computer Methods and Programs in Biomedicine*, vol. 108, no. 1, pp. 10–19, 2012.

[10] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge & Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[11] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada, December 2015, pp. 1693–1701.

[12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] B. Kemp, A. H. Zwinderman, B. Tuk, H. A. Kamphuisen, and J. J. Oberye, "Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the EEG," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 9, pp. 1185–1194, 2000.

[15] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, October 2014, pp. 1746–1751.

[16] S. Ram, H. Seirawan, S. K. Kumar, and G. T. Clark, "Prevalence and impact of sleep disorders and sleep habits in the united states," *Sleep and Breathing*, vol. 14, no. 1, pp. 63–70, 2010.

[17] A. Rechtschaffen, "A manual of standardized terminology, techniques and scoring systems for sleep stages of human subjects," *National Institute of Health*, vol. 1, pp. 204–210, 1968.

[18] R. S. Rosenberg and S. Van Hout, "The American Academy of Sleep Medicine inter-scorer reliability program: sleep stage scoring," *Journal of Clinical Sleep Medicine*, vol. 9, no. 01, pp. 81–87, 2013.

[19] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[21] A. Supratak, H. Dong, C. Wu, and Y. Guo, "DeepSleepNet: a model for automatic sleep stage scoring based on raw single-channel EEG," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 11, pp. 1998–2008, 2017.

[22] O. Tsinalis, P. M. Matthews, and Y. Guo, "Automatic sleep stage scoring using time-frequency analysis and stacked sparse autoencoders," *Annals of Biomedical Engineering*, vol. 44, no. 5, pp. 1587–1597, 2016.

[23] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, July 2015, pp. 2048–2057.

[24] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, June 2015, pp. 4694–4702.