Automatic Sleep Stage Classification Using Single-Channel EEG: Learning Sequential Features with Attention-Based Recurrent Neural Networks

Huy Phan, Fernando Andreotti, Navin Cooray, Oliver Y. Chén, and Maarten De Vos
{huy.phan,fernando.andreotti,navin.cooray,yibing.chen,maarten.devos}@eng.ox.ac.uk

Abstract—**We propose in this work a feature learning approach using deep bidirectional recurrent neural networks (RNNs) with attention mechanism for single-channel automatic sleep stage classification. We firstly decompose an EEG epoch into multiple small frames and subsequently transform them into a sequence of frame-wise feature vectors. Given the training sequences, the attention-based RNN is trained in a sequence-to-label fashion for sleep stage classification. Due to discriminative training, the network is expected to encode information of an input sequence into a high-level feature vector after the attention layer. We, therefore, treat the trained network as a feature extractor and extract these feature vectors for classification which is accomplished by a linear SVM classifier. We also propose a discriminative method to learn a filter bank with a DNN for preprocessing purpose. Filtering the frame-wise feature vectors with the learned filter bank beforehand leads to further improvement on the classification performance. The proposed approach demonstrates good performance on the Sleep-EDF dataset.**

I. Introduction

Various methods have been proposed for automatic sleep staging. Majority of them relied on hand-crafted features and conventional machine learning methods, such as Support Vector Machine (SVM) (c.f. [1] for a comprehensive review). With the rapid advance of deep learning methods, they have been recently pursued for the task, like deep autoencoders [2], deep neural networks (DNNs) [3], convolutional neural networks (CNNs) [4], [5], [6]. Despite significant improvements on different benchmark datasets have been reported by these network variants, they are incapable of modelling sequences and therefore arguably suboptimal in capturing sequential dynamics of EEG signals. It is well established that recurrent neural networks (RNNs), e.g. Long Short-Term Memory (LTSM) [7], are highly capable of sequential modelling. However, they have been often used in combination with DNNs [3] or CNNs [4] to benefit from their feature learning power. Given the sequential nature of EEG signals, to our knowledge, there is no prior work succeeding in training standalone RNNs for sequential feature learning with better or even on par performance than those learned by CNNs on the same benchmarks.

This work presents an approach that successfully learns sequential features from single-channel EEG signals for automatic sleep stage classification using a deep RNN with attention mechanism. A 30-second EEG epoch is firstly decomposed into small frames and transformed into a sequence of frame-wise feature vectors. Log-power spectral coefficients are used for frame-wise representation. An attention-based deep bidirectional RNN, as illustrated in Fig. 1, is then trained for sequence-to-label classification. Due to the discriminative training, the high-level feature vector after the attention layer of the network is expected to encode information of an entire input sequence. The feature vector are subsequently extracted and used as representation of the input sequence. The sleep stage classification is finally carried out using a linear SVM classifier. As an improvement, we propose a preprocessing step which makes use of a frequency-domain filter bank learned by a DNN to process the frame-wise feature vectors beforehand, i.e. frequency smoothing and dimension reduction. The DNN is discriminatively trained on frame-wise feature vectors to encourage the learned filter bank to emphasize the frequency subbands important for the task and attenuate those less important.

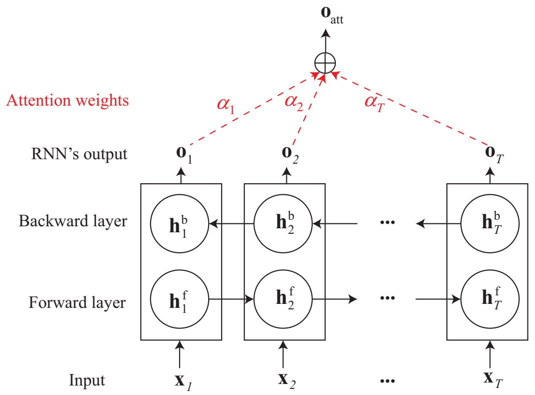

Fig. 1: Attention-based bidirectional recurrent neural network.

We show in the experiments that the proposed approach achieve good performance on the Sleep-EDF dataset using the single channel Fpz-Cz. The obtained results outperform the best results on the dataset reported in previous works, including those based on deep CNNs and deep CNNs combined with RNN layers.

II. Attention-Based Deep Recurrent Neural Network for EEG Sequence Modelling

A. *Time-Frequency Features*

As an input to the proposed RNN, a 30-second EEG epoch is necessary to be represented as a sequence of local features. To accomplish this, we decompose the EEG signal into interleaved frames of two seconds long with an overlap of 50%. This results in $T = 29$ such frames in total. Each frame is then transformed into frequency domain via 256-point discrete Fourier transform (DFT) with Hamming window, followed by logarithm scaling to obtain a log-power feature vector of size $F = 129$. Afterwards, dimension reduction and frequency smoothing are performed by filtering the log-power feature vectors with a frequency-domain triangular filter bank with $M = 20$ filters. The filters are equally spaced with an overlap of 50% as illustrated in Fig. 3 (a). Alternative to the triangular filter bank, a filter bank discriminatively learned by a DNN (c.f. Section III), can also be used for this purpose. As a result, a log-power feature vector is reduced into a M-dimensional feature vector. Eventually, we obtain a sequence of frame-wise feature vectors $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$, where $\mathbf{x} \in \mathbb{R}^M$, to represent the original EEG epoch.

HP, FA, NC, OYC, and MDV are with the Institute of Biomedical Engineering, University of Oxford, Oxford OX3 7DQ, United Kingdom.

The research was supported by the NIHR Oxford Biomedical Research Centre and Wellcome Trust under Grant 098461/Z/12/Z.

978-1-5386-3646-6/18/$31.00 ©2018 IEEE

B. Deep Bidirectional RNN

Given a sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ of T feature vectors, where $\mathbf{x} \in \mathbb{R}^M$, we aim at learning a fixed-length representation which encodes the entire input sequence using the proposed attention-based bidirectional RNN.

The proposed RNN architecture is illustrated in Fig. 1. At each recurrent layer, the network maintains two hidden layers, forward and backward. These two layers iterate over individual feature vectors of the input sequence in opposite directions and computes forward and backward sequences of hidden state vectors $\mathbf{H}^{\mathrm{f}} = (\mathbf{h}^{\mathrm{f}}_1, \ldots, \mathbf{h}^{\mathrm{f}}_T)$ and $\mathbf{H}^{\mathrm{b}} = (\mathbf{h}^{\mathrm{b}}_1, \ldots, \mathbf{h}^{\mathrm{b}}_T)$, respectively, where

$$\mathbf{h}^{\mathrm{f}}_t = \mathcal{H}(\mathbf{x}\,, \mathbf{h}^{\mathrm{f}}_{t-1}), \tag{1}$$

$$\mathbf{h}^{\mathrm{b}}_t = \mathcal{H}(\mathbf{x}\,, \mathbf{h}^{\mathrm{b}}_{t+1}),\ 1 \leq t \leq T. \tag{2}$$

In (1) and (2), $\mathcal{H}$ denotes the hidden layer function. We employ the Gated Recurrent Unit (GRU) cell [8] for both forward and backward layers mainly due to its lower computational cost compared to Long Short-Term Memory (LSTM) [7]. The GRU cell is implemented by the compound of following functions:

$$\mathbf{r}_t = \mathrm{sigm}\left(\mathbf{W}_{sr}\mathbf{s}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_r\right), \tag{3}$$

$$\mathbf{z}_t = \mathrm{sigm}\left(\mathbf{W}_{sz}\mathbf{s}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_z\right), \tag{4}$$

$$\tilde{\mathbf{h}}_t = \tanh\left(\mathbf{W}_{sh}\mathbf{s}_t + \mathbf{W}_{hh}\left(\mathbf{r}_t \odot \mathbf{h}_{t-1}\right) + \mathbf{b}_h\right), \tag{5}$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t. \tag{6}$$

In above equations, the $\mathbf{W}$ variables denote the weight matrices and the $\mathbf{b}$ variables are the biases. The $\mathbf{r}$, $\mathbf{z}$, and $\tilde{\mathbf{h}}$ variables represent the reset gate vector, the update gate vector, and the new hidden state vector candidate, respectively. The $\odot$ operator denotes the element-wise multiplication.

The network output at a time t is computed as

$$\mathbf{o}_t = \mathbf{W}_{ho}[\mathbf{h}^{\mathrm{b}}_t \oplus \mathbf{h}^{\mathrm{f}}_t] + \mathbf{b}_o, \tag{7}$$

where the $\oplus$ represents vector concatenation.

In order to construct a deep bidirectional RNN, we stack multiple RNN hidden layers on top of each other as in [9]. The forward and backward hidden state sequences of a lower recurrent layer are treated as the forward and backward input sequences for the upper layer. Assume that the network has L layers in total, the equations (1), (2), and (7) can be re-written as

$$\mathbf{h}^{\mathrm{f}}_{t,\ell} = \mathcal{H}(\mathbf{h}^{\mathrm{f}}_{t,\ell-1}\,, \mathbf{h}^{\mathrm{f}}_{t-1,,\ell}), \tag{8}$$

$$\mathbf{h}^{\mathrm{b}}_{t,\ell} = \mathcal{H}(\mathbf{h}^{\mathrm{b}}_{t,\ell-1}\,, \mathbf{h}^{\mathrm{b}}_{t+1,\ell}), \tag{9}$$

$$\mathbf{o}_t = \mathbf{W}_{ho}[\mathbf{h}^{\mathrm{b}}_{t,L} \oplus \mathbf{h}^{\mathrm{f}}_{t,L}] + \mathbf{b}_o, \tag{10}$$

respectively, where $1 \leq \ell \leq L$. Note that $\mathbf{H}^{\mathrm{f}}_0 \equiv \mathbf{H}^{\mathrm{b}}_0 \equiv \mathbf{X}$ for the first layer.

C. Attention Weights

In general, for the sequence-to-label setting, only the output vector at the last time step $\mathbf{o}_T$ is retained for classification, e.g. via a softmax layer [9]. However, it is reasonable to somehow combine the output vectors at different time steps via some weighting schemes. Intuitively, those parts of the input sequence which are discriminative for the classification task at hand should be associated with strong weights and vice versa. Ideally, these weights should be automatically learned by the network. This can be accomplished with an attention layer [10].

Formally, the attention weight α_t for the output vector $\mathbf{o}_t$ at the time step t is computed as

$$\alpha_t = \frac{\exp\left(f(\mathbf{o}_t)\right)}{\sum_{i=1}^{T}\exp\left(f(\mathbf{o}_i)\right)}. \tag{11}$$

In (11), f denotes the scoring function of the attention layer:

$$f(\mathbf{o}) = \mathbf{o}^{\mathsf{T}}\mathbf{W}_{\text{att}}, \tag{12}$$

where $\mathbf{W}_{\text{att}}$ is the trainable weight matrix of the attention layer. The attentive output feature vector is obtained as a weighting combination of the output vectors at individual time steps:

$$\mathbf{o}_{\text{att}} = \sum_{i=1}^{T}\alpha_i\mathbf{o}_i. \tag{13}$$

$\mathbf{o}_{\text{att}}$ is now considered as the high-level representation of the whole original input sequence. Finally, $\mathbf{o}_{\text{att}}$ is presented to a softmax layer for classification. The network is trained to minimize the cross-entropy error over N training samples:

$$E(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i\log(\hat{\boldsymbol{y}}_i(\boldsymbol{\theta})) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2_2. \tag{14}$$

In (14), $\boldsymbol{\theta}$, $\hat{\boldsymbol{y}}$, $\boldsymbol{y}$ denote the network parameters, the predicted posterior distribution, and the one-hot encoded groundtruth distribution, respectively. λ denotes the hyper-parameter that trades off the error terms and the ℓ_2-norm regularization term. *Dropout* [11] is also applied to the GRU cells' input and output. The network training is performed using the *Adam* optimizer [12].

D. Classification with Linear SVM

After training the network, instead of using it for classification, we treat it as a feature extractor as in [9] to extract the attentive feature vector $\mathbf{o}_{\text{att}}$ for each EEG epoch. Linear SVM is employed for classification in replacement of the softmax layer of the network. In general, SVM usually leads to better generalization in comparison to the softmax, thanks to its maximum margin property [13]. The feature vectors extracted for the training examples are used to train the SVM classifier which is subsequently employed to classify those feature vectors extracted for the test examples.

III. DNN FOR FILTER-BANK LEARNING

As an alternative to the regular triangular filter bank used for preprocessing in Section II-A, we train a tailored DNN to learn a filter bank discriminatively for this purpose. The learned filter bank is expected to emphasize the subbands that are more important for the task and attenuate those less important rather than considering them equally as in the regular triangular filter bank.

The DNN architecture proposed for filter-bank learning consists of one *filter-bank layer*, three fully-connected (FC) layers, and one softmax layer, as illustrated in Fig. 2. The filter-bank layer is actually a fully-connected layer which are enforced various constraints for filter-bank learning purpose as in [14]. The FC layers are the common nonlinear ones with ReLU activation [15].

Assume that we want to learn a filter bank with M filters. Note that M is also the number of hidden units of the filter bank layer. Given an input $\mathbf{x} \in \mathbb{R}^F$, the output of the filter-bank layer reads

$$\mathbf{h}_1 = \mathbf{x}\mathbf{W}_{\text{fb}}, \tag{15}$$

where $\mathbf{W}_{\text{fb}} \in \mathbb{R}^{F\times M}$ in (15) plays the role of the filter-bank weight matrix. Furthermore, for the learned filter bank to have the characteristics of a normal filter bank, i.e. non-negative, band limited and ordered by frequency, it is necessary to enforce constraints and re-write $\mathbf{W}_{\text{fb}}$ as

$$\mathbf{W}_{\text{fb}} = f_{+}(\mathbf{W}) \odot \mathbf{S}, \tag{16}$$

where $\mathbf{W} \in \mathbb{R}^{F\times M}$ is weight matrix that will be learned by the DNN in practice. f_{+} denotes a non-negative function to make

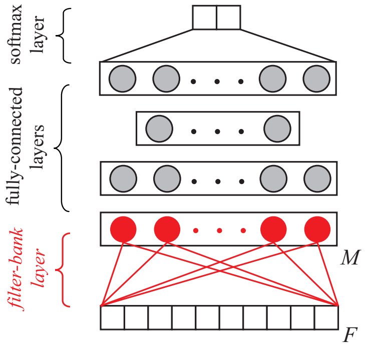

Fig. 2: Illustration of the DNN architecture for filter-bank learning.

the elements of $\mathbf{W}$ non-negative. $\mathbf{S} \in \mathbb{R}_+^{F\times M}$ is the constant non-negative matrix to enforce the filters to have limited band, regulated shape and ordered by frequency. We employ *sigmoid* for the function $f_+(x) = \frac{1}{1+\exp(-x)}$ and a linear-frequency triangular filter bank matrix for $\mathbf{S}$, as illustrated in Fig. 3 (b).

Opposing to the attention-based RNN which operates on entire sequences of frame-wise feature vectors, the DNN receives the raw (i.e. without frequency smoothing and dimension reduction) frame-wise log-power feature vectors as input. The DNN is trained to minimize the cross-entropy given in (14) (the regularization term excluded) over the training set. For training purpose, a two-second frame is labelled by the label of the 30-second epoch from which it is stemmed. Dropout is also applied to the FC layers. Fig. 3 (c) shows one of the filter banks learned in the experiments (cf. Section IV for further details).

IV. EXPERIMENTS

A. Sleep-EDF Dataset

We conducted experiments on PhysioNet's Sleep-EDF Expanded dataset [16], [17]. The dataset consists of 20 subjects in total, each of which has two PSG recordings in two subsequent date-night periods, except for subject 13. The recordings were manually scored according to the R&K standard [18], i.e. each 30-second epoch was labelled with one of eight labels {W, N1, N2, N3, N4, REM, MOVEMENT, UNKNOWN}. Two different experimental settings on this dataset has been explored by previous works. The first one (Setting 1) only considered in-bed parts of the recordings [5], [2]. The second one included both in-bed parts and 30-minute periods before and after sleep periods (Setting 2) [4]. For a proper comparison, the experiments were conducted with both settings. Moreover, as in [5], [2], [4], we merged N3 and N4 into a single stage N3 and excluded MOVEMENT and UKNOWN. Only the single channel Fpz-Cz was used in the experiments.

B. Experimental Setup

Leave-one-subject-out cross validation was performed. At each iteration, data of all 19 training subjects was used to train the filter-bank-learning DNN. Differently, for the RNN, 4 out of 19 training subjects were left out for validation while the remaining 15 subjects were used for training. During training the RNN network yielding the best overall accuracy on the validation set was retained for testing. The evaluation is based on average performance in terms of overall accuracy, macro F1-score (MF), and kappa index (κ). F1-scores of individual classes will also be reported.

C. Parameters

The parameters of the attention-based RNN and the filter-bank-learning DNN are shown in Tables I (a) and (b), respectively. The networks were implemented using the *Tensorflow* framework. Both of them were trained for 200 epochs with a batch size of 200. During training, we sampled a training batch so that there is always an equal number of samples for all classes. The learning rate was commonly fixed to 10^{-4}.

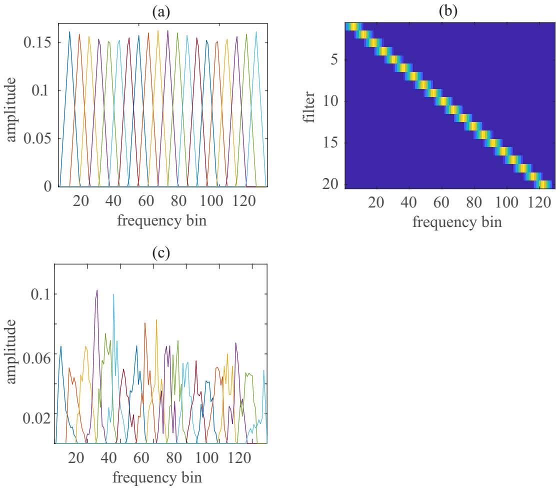

Fig. 3: (a) A linear-frequency triangular filter bank with $M = 20$ filters, (b) the corresponding shape matrix $\mathbf{S}$, (c) a DNN-learned filter bank with $M = 20$ filters.

The linear SVMs were trained using the *LIBLINEAR* library [19]. The features were normalized to the range of $[-1, 1]$ beforehand and the hyper-parameter C of the SVMs was fixed to 10^{-3}.

D. Experimental Results

We show in Table II the performances obtained by our systems as well as those results reported in previous works which are based on deep networks [2], [5], [4]. When the attention-based RNNs are used directly for classification (i.e. with their softmax layers), we denote them as *ARNN1* and *ARNN2* where the number indicates whether the regular triangular filter bank or the one learned with the DNN are used for preprocessing, respectively. Moreover, when linear SVMs are employed for classification, we denote the corresponding systems as *ARNN1-SVM* and *ARNN2-SVM*. Regarding the previous works, the best performances on the dataset were reported in [2] and [4] for Setting 1 and 2, respectively. The former (i.e. Autoencoder in Table II) relies on deep autoencoders with a handful of hand-crafted features. The later (i.e. DeepSleepNet in Table II) makes use of a deep residual network in which RNN layers are stacked on top of convolutional layers to leverage both their sequential modelling capability and feature learning power. Note that two different EEG channels were exploited for this system, Pz-Oz in DeepSleepNet1 and Fpz-Cz in DeepSleepNet2.

On one hand, the benefits of the DNN-learned filter bank on the

TABLE I: Parameters of the proposed networks: (a) attention-based deep RNN, (b) filter-bank-learning DNN.

(a)

Parameter	**Value**
The number of layers L	2
Size of hidden state vector	256
Size of the attention weights	96
Dropout rate	0.2
Regularization parameter λ	10^{-4}

(b)

Layer	**Size**	**Dropout**
FC 1	512	0.2
FC 2	256	0.2
FC 3	512	0.2

TABLE II: Performances obtained by different approaches on the Sleep-EDF dataset.

		Overall metrics			Per-class F1-score				
		Acc	MF1	κ	W	N1	N2	N3	REM
Setting 1	Deep CNN [5]	74.8	69.8	–	65.4	43.7	80.6	84.9	74.5
	Autoencoder [2]	78.9	73.7	–	71.6	47.0	84.6	84.0	81.4
	ARNN1	76.3	69.5	0.67	**75.2**	34.2	83.1	81.3	74.0
	ARNN1-SVM	78.3	69.1	0.69	**76.2**	27.6	**84.9**	81.6	75.2
	ARNN2	77.3	70.1	0.68	**75.1**	32.0	84.4	**85.0**	74.0
	ARNN2-SVM	**79.1**	69.8	0.70	**75.5**	27.3	**86.0**	**85.6**	74.8
Setting 2	DeepSleepNet1 [4]	79.8	73.1	0.72	88.1	37.0	82.7	77.3	80.3
	DeepSleepNet2 [4]	82.0	76.9	0.76	84.7	46.6	85.9	84.8	82.4
	ARNN1	75.3	67.3	0.67	87.5	31.8	78.5	68.6	70.0
	ARNN1-SVM	77.6	66.6	0.69	88.0	23.3	81.6	68.8	71.4
	ARNN2	80.7	73.2	0.74	**89.9**	33.6	84.8	**84.8**	72.7
	ARNN2-SVM	**82.5**	72.0	**0.76**	**91.5**	23.8	**86.1**	**85.3**	73.5

performances can be clearly seen from Table II. On Setting 1, using the DNN-learned filter bank leads to absolute improvements of 1.0%, 0.6%, and 0.01 in overall accuracy, MF1, and κ, respectively. The corresponding gains on Setting 2 reach 5.4%, 6.3%, and 0.07. Note that we utilized the filter banks learned in Setting 1 for the experiments in Setting 2 rather than training them from scratch. We expect that training them from scratch would result in filter banks with similar impulse responses. On another hand, the positive effects of using SVM in replacement of softmax can also be seen. Improvements are obtained by *ARNN1-SVM* and *ARNN2-SVM* over *ARNN1* and *ARNN2* on both Setting 1 and 2. For instance, using *ARNN2-SVM* results in absolute gains of 1.7% and 1.8% in overall accuracy over *ARNN2* in Setting 1 and 2, respectively.

Moreover, the performances achieved by the proposed approach are better than the best results reported by other counterparts. Specifically, *ARNN2-SVM* outperforms Autoencoder [2] in Setting 1 with an absolute gain of 0.2% and surpasses DeepSleepNet2 [4] in Setting 2 with a margin of 0.5%. Improvements on individual classes, i.e. W, N2, and N3, can also be seen. However, *ARNN2-SVM* produces a lower MF1 compared to Autoencoder and DeepSleepNet2 due to its inferior accuracy on other two stages, especially N1 which is in general hard to recognize, due to similarities with other stages and generally infrequent. The imbalance of the data is also likely a cause which provides an opportunity for further study to mitigate its effect. The confusion matrices in Table III gives insight into the classification details. Nevertheless, these results indicate that the RNNs with their sequence modelling capability are promising in capturing the sequential nature of sleep recorded from EEG signals for automatic sleep staging. Furthermore, they can work standalone rather than in conjunction with other network variants, such as CNNs [4].

V. CONCLUSIONS

We propose an attention-based RNN to learn sequential features from EEG signals for automatic sleep staging. An EEG signal is transformed into a sequence of frame-wise feature vectors which are then preprocessed with either a regular triangular filter bank or a filter bank learned with a DNN. The temporal patterns of the input sequence are then encoded by two GRU-based bidirectional layers combined with an attention layer to form the high-level feature vector which represents the input sequence. The classification is finally accomplished by linear SVMs using these high-level feature vectors. We demonstrate good performance obtained by the proposed approach on the Sleep-EDF dataset. This implies that the strong sequential modelling capability of RNNs is potential in describing the sequential nature of sleep, as demonstrated by this application in automatic sleep stage classification.

TABLE III: Confusion matrices of *ARNN2-SVM* on two experimental settings of the EDF-Sleep dataset.

		Prediction				
		W	N1	N2	N3	REM
Setting 1 Groundtruth	W	3585	280	168	48	428
	N1	532	555	674	9	992
	N2	438	182	15159	704	1094
	N3	98	0	703	4753	37
	REM	332	282	966	7	6124
Setting 2 Groundtruth	W	11583	227	168	67	473
	N1	635	461	674	12	997
	N2	262	137	15260	641	1299
	N3	114	4	742	4728	41
	REM	330	269	991	5	6116

REFERENCES

[1] R. Boostania, F. Karimzadeha, and M. Nami, "A comparative review on sleep stage classification methods in patients and healthy individuals," *Comput. Methods Programs Biomed.*, pp. 77–91, 2017.

[2] O. Tsinalis, P. M. Matthews, and Y. Guo, "Automatic sleep stage scoring using time-frequency analysis and stacked sparse autoencoders," *Ann. Biomed. Eng.*, vol. 44, no. 5, pp. 1587–1597, 2016.

[3] H. Dong, A. Supratak, W. Pan, C. Wu, P. M. Matthews, and Y. Guo, "Mixed neural network approach for temporal sleep stage classification," *IEEE Trans. Neural. Syst. Rehabil. Eng.*, 2017.

[4] A. Supratak, H. Dong, C. Wu, and Y. Guo, "Deepsleepnet: A model for automatic sleep stage scoring based on raw single-channel eeg," *IEEE Trans. Neural. Syst. Rehabil. Eng.*, vol. 25, no. 11, pp. 1998–2008, 2017.

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

[6] K. Mikkelsen and M. de Vos, "Personalizing deep learning models for automatic sleep staging," *arXiv:1801.02645*, 2018.

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

[8] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.

[9] H. Phan, P. Koch, F. Katzberg, M. Maass, R. Mazur, and A. Mertins, "Audio scene classification with deep recurrent neural networks," in *Proc. INTERSPEECH*, 2017, number 3043–3047.

[10] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv:1508.04025*, 2015.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *JMLR*, vol. 15, pp. 1929–1958, 2014.

[12] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *Proc. ICLR*, 2015, number 1-13.

[13] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. COLT*, 1992, pp. 144–152.

[14] H. Yu, Z.-H. Tan, Y. Zhang, Z. Ma, and J. Guo, "DNN filter bank cepstral coefficients for spoofing detection," *IEEE Access*, vol. 5, pp. 4779–4787, 2017.

[15] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. AISTATS*, 2011, pp. 315–323.

[16] B. Kemp, A. H. Zwinderman, B. Tuk, H. A. C. Kamphuisen, and J. J. L. Oberye, "Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the EEG," *IEEE Trans. Biomed. Eng.*, vol. 47, no. 9, pp. 1185–1194.

[17] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. 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, pp. e215–e220, 2000.

[18] J. A. Hobson, "A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects," *Electroencephalogr. Clin. Neurophysiol.*, vol. 26, no. 6, pp. 644, 1969.

[19] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: a library for large linear classification," *JMLR*, vol. 9, pp. 1871–1874, 2008.