Published in final edited form as:
IEEE Trans Neural Syst Rehabil Eng. 2019 March ; 27(3): 400–410. doi:10.1109/TNSRE.2019.2896659.

SeqSleepNet: End-to-End Hierarchical Recurrent Neural Network for Sequence-to-Sequence Automatic Sleep Staging

Huy Phan*
School of Computing, University of Kent, Chatham Maritime, Kent ME4 4AG, United Kingdom and the Institute of Biomedical Engineering, University of Oxford, Oxford OX3 7DQ, United Kingdom

Fernando Andreotti, **Navin Cooray**, **Oliver Y. Chén**, and **Maarten De Vos**
Institute of Biomedical Engineering, University of Oxford, Oxford OX3 7DQ, United Kingdom

Abstract

Automatic sleep staging has been often treated as a simple classification problem that aims at determining the label of individual target polysomnography (PSG) epochs one at a time. In this work, we tackle the task as a sequence-to-sequence classification problem that receives a sequence of multiple epochs as input and classifies all of their labels at once. For this purpose, we propose a hierarchical recurrent neural network named SeqSleepNet1. At the epoch processing level, the network consists of a filterbank layer tailored to learn frequency-domain filters for preprocessing and an attention-based recurrent layer designed for short-term sequential modelling. At the sequence processing level, a recurrent layer placed on top of the learned epoch-wise features for long-term modelling of sequential epochs. The classification is then carried out on the output vectors at every time step of the top recurrent layer to produce the sequence of output labels. Despite being hierarchical, we present a strategy to train the network in an end-to-end fashion. We show that the proposed network outperforms state-of-the-art approaches, achieving an overall accuracy, macro F1-score, and Cohen's kappa of 87.1%, 83.3%, and 0.815 on a publicly available dataset with 200 subjects.

Index Terms

automatic sleep staging; hierarchical recurrent neural networks; end-to-end; sequence-to-sequence

I Introduction

Humans spend around one-third of their lives sleeping, this process is crucial to protect the mental and physical health of an individual [1]. Sleep disorders are becoming an alarmingly common health problem, affecting millions of people worldwide. A survey conducted in the US between 1999 and 2004 reveals that 50-70 million adults suffer from over 70 different sleep disorders and 60 percent of adults report having sleep problems a few nights a week or more [2], [3].

[1]Source code is available at http://github.com/pquochuy/SeqSleepNet

*Corresponding author: h.phan@kent.ac.uk.

Europe PMC Funders Author Manuscripts

Sleep scoring [4], [5] is a fundamental step in sleep assessment and diagnosis and requires the analysis of 30-second polysomnography (PSG) epochs to determine their sleep stages. In clinical environments, sleep staging is mainly performed manually by human experts following developed guidelines [4], [5]. The scoring procedure is labor-intensive, time-consuming, costly, and prone to human errors. Therefore, a large body of work aims to automate this task [6]–[15]. Furthermore, there is an growing need of home-based sleep monitoring [16]–[19] to provide scalable monitoring solutions that would benefit a greater population and provide a platform for epidemiological studies. In order to achieve this two primary ingredients are needed. First, user-friendly, comfortable, long-term capable, clinical-grade wearable Electroencephalography (EEG) devices are required. A number of such devices were developed and validated, such as in-ear EEG [18]–[20] and around-the-ear EEG [16], [21]. Second, reliable automatic sleep staging methods are equally indispensable.

In the last few years, the research community has witnessed an influx of deep learning methods used for automatic sleep staging in replacement of conventional feature-based machine learning approaches. Deep learning methods offer several advantages over the conventional ones and have been successful in numerous other domains. First, since public sleep data are rapidly growing (i.e. hundreds to thousands of subjects are becoming a norm), deep networks are efficient in handling a large amount of data by repeatedly learning from small batches of data to converge to the final model. Second, their power in learning features automatically from low-level signals makes hand-crafting several intricate features no longer necessary. Several types of deep network architectures exist and have been proposed for automatic sleep scoring: Convolutional Neural Networks (CNNs) [8], [10], [11], [13]–[15], Deep Belief Networks (DBNs) [22], Auto-encoder [23], Deep Neural Networks (DNNs), and Recurrent Neural Networks (RNNs) [24]. Combinations of different architectures, such as DNN+RNN [25] and CNN+RNN [9], [12] have also been exploited. With the deep learning methods evolving, automatic sleep staging performance has been boosted considerably as state-of-the-art results have been reported on several datasets [8], [9], [12], [13].

There are many ways to characterize existing works in automatic sleep staging, such as single-channel versus multichannel and shallow learning vs deep learning. Here, we pursuit an approach that categorizes them into classification schemes based on the number of input epochs and output labels during classification. To this end, prior works can be grouped into *one-to-one*, *many-to-one*, *one-to-many* schemes as illustrated in Fig. 1 (a)-(c), respectively. Following the one-to-one scheme, a classification model receives a single PSG epoch as input at a time and produces a single corresponding output label [14], [15], [24], [26]. Although being straightforward, this classification scheme cannot take into account the existing dependency between PSG epochs [4], [8], [27], [28]. As an extension of the one-to-one, the many-to-one scheme augments the classification of a target epoch by additionally combining it with its surrounding epochs to make a *contextual input*. This scheme has been the most widely used in prior works, not only those relying on more conventional methods [29], [30] but also modern deep neural networks [9]–[11], [13], [23], [25]. The work in [8] showed that while the *contextual input* does not always lead to performance improvement regardless of the choice of classification model, it also suffers from the modelling ambiguity and high computational overhead. The one-to-many scheme is orthogonal to the many-to-

one scheme and was recently proposed in [8] with the concept of *contextual output*. Under this scheme, a multitask model receives a single target epoch as input and jointly determines both the target label and the labels of its neighboring epochs in the contextual output. This scheme is still able to leverage the inter-epoch dependency while avoiding the limitations of the contextual input in the many-to-one-scheme. More importantly, the underlying multitask model has the capability to produce an ensemble of decisions on a certain epoch which can be then aggregated to yield a more reliable final decision [8]. However, a common drawback of both many-to-one and one-to-many schemes is that they cannot accommodate a long context, e.g. tens of epochs.

In this work, we seek to overcome this major limitation and unify all aforementioned classification schemes with the proposed many-to-many approach illustrated in Fig. 1(d). Our goal is to map an input sequence of multiple epochs to the sequence of all target labels at once. Therefore, the automatic sleep staging task is framed as a *sequence-to-sequence* classification problem. With this generalized scheme, we can circumvent disadvantages of other schemes (i.e. short context, modelling ambiguity, and computational overhead) while maintaining the one-to-many's advantage regarding the availability of decision ensemble. It should be stressed that the sequence-to-sequence problem formulated here does not simply imply a set of one-to-one mappings between one epoch in the input sequence and its corresponding label in the output sequence. In contrast, due to the inter-epoch dependency, a label in the output sequence may inherently interact with all epochs in the input sequence via some intricate relationship that need to be modelled. To accomplish sequence-to-sequence classification we present *SeqSleepNet*, a hierarchical recurrent neural network architecture. SeqSleepNet is composed of three main components: (1) parallel *filterbank layers* for preprocessing, (2) an epoch-level bidirectional RNN coupled with the attention mechanism for short-term (i.e. intra-epoch) sequential modelling, and (3) a sequence-level bidirectional RNN for long-term (i.e. inter-epoch) sequential modelling. The network is trained in an end-to-end manner. End-to-end network training is desirable in deep learning as an end-to-end network learns the global solution directly in contrast to multiple-stage training that estimates local solutions in separate stages. The power of end-to-end learning has been proven many times in various domains [31]–[36]. Moreover, end-to-end training is more convenient and elegant.

Our proposed method bears resemblance to some existing works. Learning data-driven filters with a filterbank layer has been shown to be efficient in our previous works [8], [24], [26]. However, instead of training a filterbank layer separately with a DNN, here multiple filterbank layers for multichannel input are parts of the classification network and are trained end-to-end. There also exists a few multiple-output network architectures proposed for automatic sleep staging, nevertheless, they are either limited to accommodate a long-term context [8] or need to be trained in multiple stages rather than end-to-end [9], [25]. In addition, these works used CNNs or DNNs for epoch-wise feature learning while, in the proposed SeqSleepNet, we employ a recurrent layer coupled with the attention mechanism for this purpose. Given the sequential nature of sleep data, the sequential modelling capability of RNNs [37], [38] make them potential candidates for this purpose but have been left uncharted. On one hand, we demonstrate that the sequential features learned with the attention-based recurrent layer result in a better performance than the convolutional ones. On

the other hand, using our end-to-end training strategy, we also build end-to-end variants of these multiple-output networks as baselines and show that the proposed DeepSleepNet significantly outperforms all these baselines and set state-of-the-art performance on the experimental dataset.

II Montreal Archive of Sleep Studies (MASS) Dataset

The public dataset Montreal Archive of Sleep Studies (MASS) [39] was used for evaluation. MASS is a considerably large open-source dataset which were pooled from different hospital-based sleep laboratories. It consists of whole-night recordings from 200 subjects aged between 18-76 years (97 males and 103 females), divided into five subsets (SS1 - SS5). Each epoch of the recordings was manually labelled by experts according to the AASM standard [4] (SS1 and SS3) or the R&K standard [5] (SS2, SS4, and SS5). We converted different annotations into five sleep stages {W, N1, N2, N3, and REM} as suggested in [40], [41]. Furthermore, those recordings with 20-second epochs were converted into 30-second ones by including 5-second segments before and after each epoch. In our analysis, we used the entire dataset (i.e. all five subsets), following the experimental setup suggested in [8]. Apart from an EEG channel, an EOG and EMG channel were included to complement the EEG as they have been shown to be valuable addition sources for automatic sleep staging [8], [11]–[14], [42], [43]. We adopted and studied combinations of the C4-A1 EEG, an average EOG (ROC-LOC), and an average EMG (CHIN1-CHIN2) channels in our experiments. The signals, originally sampled at 256 Hz, were downsampled to 100 Hz.

III SeqSleepNet: End-to-End Hierarchical Recurrent Neural Network

The proposed SeqSleepNet for sequence-to-sequence sleep staging is illustrated in Fig. 2. Formally, given a sequence of PSG epochs of length L represented by $(\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_L)$, the goal is to compute a sequence of outputs $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_L)$ that maximizes the conditional probability $p(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_L / \mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_L)$.

An epoch in the input sequence consisting of C channels (i.e. EEG, EOG, and EMG in this work), are firstly transformed into a time-frequency image $\mathbf{S}$ of C image channels. Parallel filterbank layers [24], [26] are tailored to learn channel-specific frequency-domain filterbanks to preprocess the input image for frequency smoothing and dimension reduction. Furthermore, after channel-specific preprocessing, all image channels are concatenated in the frequency direction to form an image $\mathbf{X}$. The image $\mathbf{X}$ itself can be interpreted as a sequence of feature vectors which correspond to the image columns. The epoch-level attention-based bidirectional RNN is then used to encode the feature vector sequence of the epoch into a fixed attentional feature vector $\bar{\mathbf{a}}$. Finally, the sequence of attentional feature vectors $\overline{\mathbf{A}} = (\bar{\mathbf{a}}_1, \bar{\mathbf{a}}_2, \ldots, \bar{\mathbf{a}}_L)$ obtained from the input epoch sequence are modelled by the sequence-level bidirectional RNN situating on top of the network hierarchy to compute the output sequence $\widehat{\mathbf{Y}} = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_L)$.

It should be noted that, in the SeqSleepNet, the filterbank layers are tied (i.e. shared parameters) between all epochs' local features (i.e. spectral image columns) and the epoch-

level attention-based bidirectional RNN layer are tied between all epochs in the input sequence.

A Time-Frequency Image Representation

The constituent signals of a 30-second PSG epoch (i.e. EEG, EOG, and EMG) are transformed into power spectra via short-time Fourier transform (STFT) with a window size of two seconds and 50% overlap. Hamming window and 256-point Fast Fourier Transform (FFT) are used. Logarithm scaling is then applied to the spectra to convert them into log-power spectra. As a result, a multi-channel image $\mathbf{S} \in \mathbb{R}^{F \times T \times C}$ is obtained where $F = 129$, $T = 29$, and $C = 3$ denote the number of frequency bins, the number of spectral columns (i.e. time indices), and the number of channels.

B Filterbank Layers

We tailor a filterbank layer for learning frequency-domain filterbanks as in our previous works [24], [26]. The learned filterbank is expected to emphasize the subbands that are more important for the task at hand and attenuate those less important. However, instead of training a separate DNN for this purpose, the filterbank layers are parts of the classification network SeqSleepNet and are learned end-to-end. Moreover, due to the different signal characteristics of EEG, EOG, and EMG, it is reasonable to learn C channel-specific filterbanks with C separate filterbank layers.

Considering the c-th filterbank layer with respect to the c-th image channel $\mathbf{S}^c \in \mathbb{R}^{F \times T}$ where $1 \leq c \leq C$ and assuming that we want to learn a frequency-domain filertbank of M filters where $M < F$, the filterbank layer in principle is a fully-connected layer of M hidden units. The weight matrix $\mathbf{W}^c \in \mathbb{R}^{F \times M}$ of this layer plays the role of the filterbank's weight matrix. Since a filterbank has characteristics of being non-negative, band-limited, and ordered in frequency, it is necessary to enforce the following constraints [44] for the learned filterbank to have these characteristics:

$$\mathbf{W}_{\text{fb}}^c = f_+(\mathbf{W}) \odot \mathbf{T}. \quad (1)$$

Here, f_+ denotes a non-negative function to make the elements of $\mathbf{W}$ non-negative, in this study the *sigmoid* function is adopted. $\mathbf{T} \in \mathbb{R}_+^{F \times M}$ is the constant non-negative matrix to enforce the filters to have limited band, regulated shape and ordered by frequency. Similar to [26], we employ a linear-frequency triangular filterbank matrix for $\mathbf{T}$. The $\odot$ operator denotes the element-wise multiplication.

Presenting the image $\mathbf{S}^c$ to the filterbank layer, we obtained an output image $\mathbf{X}^c \in \mathbb{R}^{M \times T}$ given by

$$\mathbf{X}^c = {\mathbf{W}_{\text{fb}}^c}^\top \mathbf{S}^c. \quad (2)$$

All together, filtering the C-channel input image $\mathbf{S} \in \mathbb{R}^{F \times T \times C}$ in frequency direction with C filterbank layers results in the C-channel output image $\mathbf{X} \in \mathbb{R}^{M \times T \times C}$ which has smaller size in frequency dimension. Eventually, we concatenate the image channels of $\mathbf{X}$ in frequency direction to make $\mathbf{X}$ a single-channel image of size $MC \times T$.

C Short-term Sequential Modelling

Many approaches to extract features that represent an epoch exist. Apart from a large body of hand-crafted features [29], automatic feature learning with deep learning approaches are becoming more common [9]–[12], [14], [22]–[26], [45]–[47]. Here, we employ a bidirectional RNN coupled with the attention mechanism [48], [49] to learn sequential features for epoch representation. Due to the RNN's sequential modelling capability, it is expected to capture temporal dynamics of input signals to produce good features [24].

For convenience, we interpret the image $\mathbf{X}$ after the filterbank layers as a sequence of T feature vectors $\mathbf{X} \equiv (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_T)$ where each $\mathrm{x}_t \in \mathbb{R}^{MC}$, $1 \leq t \leq T$, is the image column at time index t. We then aim to read the sequence of feature vectors into a single feature vector using the attention-based bidirectional RNN.

The forward and backward recurrent layers of the RNN iterate over individual feature vectors of the sequence in opposite directions and compute forward and backward sequences of hidden state vectors $\mathbf{H}^{\mathrm{f}} = (\mathbf{h}_1^{\mathrm{f}}, \mathbf{h}_2^{\mathrm{f}}, \ldots, \mathbf{h}_T^{\mathrm{f}},)$ and $\mathbf{H}^{\mathrm{b}} = (\mathbf{h}_1^{\mathrm{b}}, \mathbf{h}_2^{\mathrm{b}}, \ldots, \mathbf{h}_T^{\mathrm{b}},)$ respectively, where

$$\mathbf{h}_t^{\mathrm{f}} = \mathscr{H}(\mathbf{x}_t, \mathbf{h}_{t-1}^{\mathrm{f}}), \quad (3)$$

$$\mathbf{h}_t^{\mathrm{b}} = \mathscr{H}(\mathbf{x}_t, \mathbf{h}_{t+1}^{\mathrm{b}}), \quad 1 \leq t \leq T. \quad (4)$$

In (3) and (4), $\mathscr{H}$ denotes the hidden layer function. Long Short-Term Memory (LSTM) [37] and Gated Recurrent Unit (GRU) cell [38] are most commonly used for $\mathscr{H}$. LSTM cell and GRU cell have been shown to perform comparably on many machine learning tasks, however, the latter has less parameters and is therefore more computational-efficient than the former [50]. Here, we employ the latter which is implemented by the following functions:

$$\mathbf{r}_t = \mathrm{sigm}\left(\mathbf{W}_{sr}\mathbf{s}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_r\right), \quad (5)$$

$$\mathbf{z}_t = \mathrm{sigm}\left(\mathbf{W}_{sz}\mathbf{s}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_z\right), \quad (6)$$

$$\overline{\mathbf{h}}_t = \tanh\left(\mathbf{W}_{sh}\mathbf{s}_t + \mathbf{W}_{hh}\left(\mathbf{r}_t \odot \mathbf{h}_{t-1}\right) + \mathbf{b}_h\right), \quad (7)$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \overline{\mathbf{h}}_t, \quad (8)$$

where the $\mathbf{W}$ variables denote the weight matrices and the $\mathbf{b}$ variables are the biases. The $\mathbf{r}$, $\mathbf{z}$, and $\overline{\mathbf{h}}$ variables represent the reset gate vector, the update gate vector, and the new hidden state vector candidate, respectively.

The RNN produces the sequence of output vectors $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_T)$ where $\mathbf{a}_t$ is computed as

$$\mathbf{a}_t = \mathbf{W}_{ha}\left[\mathbf{h}_t^{\mathrm{b}} \oplus \mathbf{h}_t^{\mathrm{f}}\right] + \mathbf{b}_a, \quad (9)$$

where $\oplus$ represents vector concatenation.

The attention layer [48], [49] is then used to learn a weighting vector to combine these output vectors at different time steps into a single feature vector. The rationale is that those parts of the sequence which are more informative should be associated with strong weights and vice versa. Formally, the attention weight α_t at the time index t is computed as

$$\alpha_t = \frac{\exp\left(f(\mathbf{a}_t)\right)}{\sum_{i=1}^{T} \exp\left(f(\mathbf{a}_i)\right)}. \quad (10)$$

Here, f denotes the scoring function of the attention layer and is given by

$$f(\mathbf{a}) = \mathbf{a}^{\top}\mathbf{W}_{\mathrm{att}}, \quad (11)$$

where $\mathbf{W}_{\mathrm{att}}$ is the trainable weight matrix. The attentional feature vector $\overline{\mathbf{a}}$ is obtained as a weighting combination of the recurrent output vectors:

$$\overline{\mathbf{a}} = \sum_{t=1}^{T} \alpha_t \mathbf{a}_t. \quad (12)$$

The attentional feature vector $\overline{\mathbf{a}}$ is used as the representation of the PSG epoch in the next sequence-level modelling.

D Long-term Sequential Modelling

Processing the input sequence $(\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_L)$ with the filterbank layers in Section III-B and the attention-based bidirectional RNN layer in Section III-C results in a sequence of attentional feature vectors $\overline{\mathbf{A}} = \left(\overline{\mathbf{a}}_1, \overline{\mathbf{a}}_2, \ldots, \overline{\mathbf{a}}_L\right)$ where $\overline{\mathbf{a}}_l$, $1 \leq l \leq L$, is given in (12). The sequence-level bidirectional RNN is then used to model the sequence of epoch-wise feature

vectors to encode long-term sequential information across epochs. Similar to the bidirectional RNN used for short-term sequential modelling in Section III-C, its forward and backward sequences of hidden state vectors $\widetilde{\mathbf{H}}^{\mathrm{f}} = (\widetilde{\mathbf{h}}_1^{\mathrm{f}}, \widetilde{\mathbf{h}}_2^{\mathrm{f}}, \ldots, \widetilde{\mathbf{h}}_L^{\mathrm{f}})$ and $\widetilde{\mathbf{H}}^{\mathrm{b}} = (\widetilde{\mathbf{h}}_1^{\mathrm{b}}, \widetilde{\mathbf{h}}_2^{\mathrm{b}}, \ldots, \widetilde{\mathbf{h}}_L^{\mathrm{b}})$ are computed using (3) and (4) with $\overline{\mathbf{A}} = (\overline{\mathbf{a}}_1, \overline{\mathbf{a}}_2, \ldots, \overline{\mathbf{a}}_L)$ now playing the role of the input sequence. Again, GRU cells [38] are used for its forward and backward recurrent layers.

The sequence of output vectors $\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_L)$ is then obtained where $\mathbf{o}_l$, $1 \leq l \leq L$, is computed as

$$\mathbf{o}_l = \widetilde{\mathbf{W}}_{ho}\left[\widetilde{\mathbf{h}}_l^{\mathrm{b}} \oplus \widetilde{\mathbf{h}}_l^{\mathrm{f}}\right] + \widetilde{\mathbf{b}}_o . \quad (13)$$

Each output vector ol is presented to a softmax layer for classification to produce the sequence of classification outputs $\widehat{\mathbf{Y}} = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_L)$, where $\hat{\mathbf{y}}_l$ is a output probability distribution over all sleep stages.

E Sequence Loss

In the proposed sequence-to-sequence setting, we want to penalize the network for misclassification of any element of an input sequence. Given the input sequence ($\mathbf{S}_1$, $\mathbf{S}_2, \ldots, \mathbf{S}_L$) with the ground-truth one-hot encoding vectors ($\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_L$) and the corresponding sequence of classification outputs $(\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_L)$, the sequence loss reads as follows (note that the sequence loss E^s is normalized by the sequence length L):

$$E^s(\boldsymbol{\theta}) = -\frac{1}{L}\sum_{l=1}^{L} \mathbf{y}_l \log(\hat{\mathbf{y}}_l(\boldsymbol{\theta})) . \quad (14)$$

The network is trained to minimize the sequence loss over N training sequences in the training data:

$$E(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N} E_n^s(\boldsymbol{\theta}) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2, \quad (15)$$

where E_n^s is given in (14). Here, λ denotes the hyper-parameter that trades off the error terms and the ℓ_2-norm regularization term.

F End-to-End Training Details

In the proposed SeqSleepNet, the input unit of a filterbank layer is a spectral column of an epoch's time-frequency image, that of the epoch-level attention-based bidirectional RNN is such an entire image, and that of the top sequence-level RNN is a sequence of attentional feature vectors encoding the input epoch sequence. In order to train the network end-to-end,

we adaptively manipulate the input data, i.e. folding and unfolding, at different levels of the network hierarchy.

For simplicity, let us assume the single-channel input, and therefore, the network has only one filterbank layer. Since the network, in practice, is trained with a mini batch of data at a time, assume that at each training iteration we use a minibatch of S sequences, each consists of L epochs. For a recall, each epoch itself is represented by an time-frequency image of size $T \times F$ (cf. Section III-A) which will be interpreted as a sequence of T image columns when necessary. We firstly unfold the S input sequences to make a set of $S \times L \times T$ image columns, each of size F, to present to the filterbank layer. After the filterbank layer, we obtain a set of $S \times L \times T$ image columns but now each has a size of M. This set of image columns are then folded to form a set of $S \times L$ images, each of size $T \times M$, to feed into the epoch-level attention-based bidirectional RNN. This layer encodes each image into an attentional feature vector, resulting in a set of $S \times L$ such feature vectors. Eventually, this set of feature vectors are folded into a set of S sequences, each consists of L attentional feature vectors, to present to the sequence-level bidirectional RNN for sequence-to-sequence classification.

IV Ensemble of Decisions and Probabilistic Aggregation

Since SeqSleepNet is a multiple-output network, advancing the input sequence of size L by one epoch when evaluating it on a test recording will result in an ensemble of L decisions at every epoch (except those at the recording's ends). Fusing this decision ensemble leads to a final decision which are usually better than individual ones [8].

We use the multiplicative aggregation scheme which are shown in [8] to be efficient for this purpose. The final posterior probability of a sleep stage $y_t \in \mathcal{C} = \{\text{W, N1, N2, N3, REM}\}$ at a time index t is given by

$$P(y_t) = \frac{1}{L} \prod_{i=t-L+1}^{t} P(y_t | \mathcal{S}_i). \quad (16)$$

where $\mathcal{S}_i = (\mathbf{S}_i, \mathbf{S}_{i+1}, \ldots, \mathbf{S}_{L-1})$ is the epoch sequence starting at i. In order to avoid possible numerical problems when the ensemble size is large, it is necessary to carry out the aggregation in the logarithm domain. The equation (16) is then re-written as

$$\log P(y_t) = \frac{1}{L} \sum_{i=t-L+1}^{t} \log P(y_t | \mathcal{S}_i). \quad (17)$$

Eventually, the predicted label $\hat{y}_t$ is determined by likelihood maximization:

$$\hat{y}_t = \underset{y_t}{\operatorname{argmax}} \log P(y_t) \text{ for } y_t \in \mathscr{L}. \quad (18)$$

V Experiments

A Experimental Setup

We conducted 20-fold cross validation on the MASS dataset. At each iteration, 200 subjects were split into training, validation, and test set with 180, 10, and 10 subjects, respectively. During training, we evaluated the network after every 100 training steps and the one yielded the best overall accuracy on the validation set was retained for evaluation. The outputs of 20 cross-validation folds were pooled and considered as a whole for computing the sleep staging performance.

B Network Parameters

The network was implemented using *TensorFlow v1.3.0* framework [51]. The network parameters are shown in Table I. Particularly, we experimented with different sequence length of {10, 20, 30} epochs, which is equivalent to {5, 10, 15} minutes, to study its influence. The network was trained for 10 epochs with a minibatch size of 32 sequences. The sequences were sampled from the PSG recordings with a maximum overlapping (i.e. L – 1 epochs), in this way, we generated all possible epoch sequences from the training data.

Beside ℓ_2-norm regularization in (15), dropout [52] was employed for further regularization. Recurrent batch normalization [53] was also integrated to the GRU cell to improve its convergence. The network training was performed using *Adam* optimizer [54] with a learning rate of 10^{-4}.

C Baseline Networks

In order to assess the efficiency of the proposed SeqSleepNet, apart from existing works, we developed three novel end-to-end baseline networks[2] for comparison:

End-to-end ARNN (E2E-ARNN)—As illustrated in Fig. 3a, E2E-ARNN is the combination of the filterbank layers and the epoch-level attention-based bidirectional RNN of the proposed SeqSleepNet, and therefore, is purposed for short-term sequential modelling. The objective is to assess the efficacy of the attention-based bidirectional RNN in epoch-wise feature learning. This baseline follows the standard one-to-one classification scheme, receiving a single epoch as input and outputting the corresponding sleep stage. The classification is accomplished by presenting the attentional output to a softmax layer. The network was trained with the standard cross-entropy loss. A similar attention-based bidirectional RNN was demonstrated to achieve good performance on a single-channel EEG setting in our previous work [26]. However, here the filterbank learning and the sleep stage classification are jointly learned in an end-to-end manner. We used similar parameters as the

[2]Source code is available at http://github.com/pquochuy/SeqSleepNet

Europe PMC Funders Author Manuscripts

SeqSleepNet's epoch-level processing block, except for the size of the attention weights which was set to 32. In addition, the network was trained for 20 epochs and was validated every 500 steps during training.

Multitask E2E-ARNN—Inspired by multitask networks for sleep staging in [8], this multitask network extends the E2E-ARNN baseline above to jointly determine the label of the input epoch and to predict the labels of its neighboring epochs. Therefore, this multiple-output baseline offers ensemble of decisions which was aggregated using the method described in Section IV. We used a context output size of 3 as in [8].

End-to-end DeepSleepNet (E2E-DeepSleepNet)—Supratak *et al.* [9] recently proposed DeepSleepNet and reported good performance on the MASS's subset SS3 with 62 subjects. This network comprises a deep CNN for epoch-wise feature learning topped up with a deep bidirectional RNN for capturing stage transitions. As described in [9], these two parts were trained in two separate stages to yield good performance. Here, we developed an *end-to-end* variant of DeepSleepNet, illustrated in Fig. 3b, and trained the model end-to-end using a similar strategy described in Section III-F. We will show that E2E-DeepSleepNet achieves a comparable performance (if not better) as that reported in [9]. The network parameters were kept as in the original version [9], however, we experimented with a sequence length of {10, 20, 30} epochs to have a comprehensive comparison with the proposed SeqSleepNet.

D Experimental Results

1) Sleep stage classification performance—We show in Table II a comprehensive performance comparison of the proposed SeqSleepNet, the developed baselines, as well as published results on the MASS dataset. We report performance of a system using overall metrics, including accuracy, macro F1-score (MF1), Cohen's kappa (κ), sensitivity, and specificity. Performance on individual sleep stages are also assessed via class-wise sensitivity and selectivity as recommended in [40]. The systems are grouped into single-output or multiple-output to ease the interpretation.

Impact of short-term sequential modelling. The efficiency of short-term sequential modelling is highlighted by the superior performance of the E2E-ARNN baseline over those of the single-output systems. Compared to the best single-output CNN opponent (i.e. 1-max CNN [8]) on the entire MASS dataset, the E2E-ARNN baseline yields improvements of 0.9% on overall accuracy. It also largely outperforms other single-output CNN architectures by 2.9% to 5.7%. Performance gains can also be consistently seen on other metrics. It should be highlighted that the E2E-ARNN baseline adheres to the very standard one-to-one classification setup and does not make use of contextual input with multiple epochs as in many other CNN opponents, such as those proposed by Chambon *et al.* [13] and Tsinalis *et al.* [10].

Single output vs multiple output. Comparing the multi-output systems, the proposed SeqSleepNet outperforms other systems and set state-of-the-art performance on the MASS dataset with an overall accuracy, MF1, and κ of 87.1%, 83.3%, and 0.815, respectively. On the entire MASS dataset, it leads to an accuracy gain of 0.7% absolute over the E2E-

DeepSleepNet baseline which is the best competitor. Given that the top recurrent layers behave similarly on two networks (although SeqSleepNet has only one recurrent layer on the sequence level as well as smaller size of hidden state vectors), the improvement is likely due to the good epoch-wise sequential features learned by the epoch-level processing block of SeqSleepNet. On individual sleep stages, SeqSleepNet and the E2E-DeepSleepNet are comparable for Wake and N2 while the former shows its prominence on N1 which is usually very challenging to be recognized due to its similar characteristics to other stages and its low prevalence. Interestingly, in REM, SeqSleepNet is superior on sensitivity but inferior on selectivity compared to E2E-DeepSleepNet. This result suggests that SeqSleepNet is less conservative than E2E-DeepSleepNet on recognizing REM, i.e. it recognizes more but slightly lower-fidelity REM epochs. The opposite is observed on N3. Regarding the family of multitask networks, although the advantage of contextual output [8] is reflected by the improvement of these networks, i.e. the multitask CNN and the M-E2E-ARNN baseline, over their single-output peers, the limit of the contextual output size [8] makes their performance incomparable to those of the SeqSleepNet and the E2E-DeepSleepNet both of which can accommodate a much longer context, thanks to the capability of their sequence-level recurrent layers.

Benefits of long-term sequential modelling. The performance boost made by the proposed SeqSleepNet and the E2E-DeepSleepNet over their single-output counterparts also shed light into the power of long-term sequential modelling for automatic sleep staging. Averaged over all experimented sequence lengths, an accuracy gain of 3.4% absolute is obtained by SeqSleepNet over the E2E-ARNN baseline. Likewise, an average accuracy improvement of 5.6% yielded by the E2E-DeepSleepNet baseline over its bare CNN version (i.e. DeepSleepNet1 [8]) can also be seen. Previous works, e.g. Supratak *et al.* [9] and Dong *et al.* [25] also presented a similar finding on the MASS subset SS3. However, the state-of-the-art performance of the proposed SeqSleepNet and the developed E2E-DeepSleepNet are obtained with end-to-end training, implying the unnecessity of multi-stage training [9], [25].

In order to reveal the cause of improvement made by longterm sequential modelling, we further examine its effects on performances of individual classes. C_1, C_2 and $C_1 - C_2$ are shown in Fig. 4. To this end, we computed the confusion matrix of the proposed SeqSleepNet with the sequence length of $L = 20$ (denoted as C_1), the confusion matrix of the E2E-ARNN baseline (denoted as C_2), and inspect the difference between them, i.e. $C_1 - C_2$. In $C_1 - C_2$, both positive diagonal entries and negative off-diagonal entries indicate improvements of SeqSleepNet over the E2E-ARNN baseline. It turns out that, long-term sequential modelling results in significant improvement on N1 with its accuracy boosted by 17.2% while subtle influence is seen on other sleep stages. This achieved accuracy on the challenging N1 stage is also better than those reported in previous works [8], [9], [13], [25]. These results suggests that long-term sequential modelling is more important than specific changes in the sleep stages.

2) Hypnogram—Fig. 5 further shows the output hypnogram and the posterior probability distribution per stage of sleep of a subject of the MASS dataset (subject 22 of subset SS1). It can be seen that the output hypnogram aligns very well with the corresponding ground truth. Often, the network makes errors at the short stage transition epochs. More specifically, on

the entire MASS dataset, out of misclassified epochs made by SeqSleepNet-20, 44.0% are transitioning and the rest 56.0% are non-transitioning. However, when we inspected the transitioning set (constituting 16.6% of the data) and the non-transitioning set (constituting 83.4% of the data) seperately, an error rate of 34.5% is seen on the former whereas that of the latter is four times lower, only 8.7%. This result suggests that the transitioning epochs are much harder to correctly classified compared to the non-transitioning ones. The rationale is that the transitioning epochs often contain information of two or three sleep stages, not to mention that the way we converted 20-second epochs to 30-second ones (cf. Section II) makes the stage overlap even worse. As a result, these present stages are active as indicated in the probability distribution in Fig. 5, however, we had to pick one of them as the final discrete output label for the sleep staging task.

3) Influence of the sequence length and the network's depth—It can be seen from the results in Table II that the sequence length equal or greater than 10 has minimal impact on the network performance. This observation is generalized for both SeqSleepNet and the E2E-DeepSleepNet as their accuracies vary in a negligible margin of 0.1% when $L = \{10, 20, 30\}$.

We carried out an additional experiment to study the influence of the deepness of SeqSleepNet's recurrent layers. We constructed the SeqSleepNet with two layers for both its epoch-level and sequence-level recurrent layers. A deep RNN was formed by stacking the GRU cells one on another as in [24], [55]. The overall accuracy of this network is shown in Table III alongside that of the SeqSleepNet which has recurrent depth of 1. The results reveal that increasing the number of recurrent layers does not change the network's accuracy when the sequence length is sufficiently large, i.e. $L = 20, 30$. With $L = 10$, an accuracy drop of 0.2% is noticeable. A possible explanation is that, with short sequence length, the stronger network with the recurrent depth of 2 is more prone to overfitting than the simpler one with the recurrent depth of 1. This effect is not observed with larger sequence lengths as heavier multitasking helps to regularize the networks better.

4) Visualization of the learned attention weights—To shed light on how the SeqSleepNet has picked up features to distinguish one sleep stage from others, Fig. 6 shows the attention weights for five specific epochs of different sleep stages. As expected, for the Wake epoch, the attention weights are particularly large in the region of high brain activities and muscle tone which are common characteristics discriminating Wake against other sleep stages. Similarly, for the REM epoch, more attention weights are put on ocular activities which are REM representative. Interestingly, attention layers also capture typical features of the N2 and N3 epoch as stronger weights are seen with occurrences of K-complex and slow brain waves, respectively.

VI Discussion

With the good performance demonstrated, the proposed SeqSleepNet has the potential to automate and replace manual sleep scoring [4], [5]. Although SeqSleepNet's overall performance is just approximately 1% better than that of the runnerup DeepSleepNet, it is worth noticing that this improvement is not evenly distributed over all sleep stages (cf. Table

II). While the networks perform more or less comparably on some stages (e.g. N2 and Wake), SeqSleepNet significantly outperforms DeepSleepNet on other stages (e.g. N1 and REM). This result might also be clinically meaningful as performing well on N1 and REM sleep makes SeqSleepNet potentially useful for diagnosis and assessments of many types of sleep disorders, such as narcolepsy [56] and REM-Sleep Behavior Disorder (RBD) [57]. It is unlikely that SeqSleepNet trained on the MASS dataset, a cohort of healthy subjects, would directly work well on subjects with sleep disorders due to their different sleep architectures and characteristics compared to the healthy controls. However, a SeqSleepNet pre-trained with a large healthy cohort like the MASS dataset could serve as a starting point to be finetuned for another cohort of sleep pathologies, especially when the target cohort is of small size.

SeqSleepNet also comes with some disadvantages. First, as a sequence-to-sequence model, the network needs to access entire sequences of multiple epochs to perform classification. This could delay online and realtime applications, such as sleep monitoring [16], [17]. Second, the class-wise results in Table II show opposite behaviors of SeqSleepNet and DeepSleepNet on N3 and REM. This suggests that DeepSleepNet could compensate SeqSleepNet to improve performance on these two stages. It is therefore worth exploring their possible combinations to leverage their respective advantages.

VII Conclusions

We proposed to treat automatic sleep staging as a sequence-to-sequence classification problem to jointly classify a sequence of multiple epochs at once. We then introduced a hierarchical recurrent neural network, i.e. SeqSleepNet, running on multichannel time-frequency image input to tackle this problem. The network is composed of parallel filterbank layers for preprocessing the image input, an epochlevel attention-based bidirectional RNN layer to encode sequential information of individual epochs, and a sequence-level bidirectional RNN layer to model inter-epoch sequential information. The network was trained end-to-end via dynamic folding and unfolding the input sequence at different levels of network hierarchy. We show that while sequential features learned for individual epochs by the epoch-level attention-based bidirectional RNN are more favourable than those learned by different CNN opponents, further capturing the long-term dependency between epochs by the top RNN layer leads to significant performance improvement. The proposed SeqSleepNet outperforms not only existing works but also the strong baselines developed for comparison, setting state-of-the-art performance on the entire MASS dataset.

Acknowledgement

The research was supported by the NIHR Oxford Biomedical Research Centre, Wellcome Trust (grant 098461/Z/12/Z), and the Engineering and Physical Sciences Research Council (EPSRC – grant EP/N024966/1).

References

[1]. Siegel JM. Clues to the functions of mammalian sleep. Nature. 2005; 437(27):1264–1271. [PubMed: 16251951]

[2]. Institute of Medicine. Sleep Disorders and Sleep Deprivation: An Unmet Public Health Problem. Washington DC: The National Academies Press; 2006.

Europe PMC Funders Author Manuscripts

[3]. Krieger, AC, editor. Social and Economic Dimensions of Sleep Disorders, An Issue of Sleep Medicine Clinics. Elsevier; 2017.

[4]. Iber C, et al. The AASM manual for the scoring of sleep and associated events: Rules, terminology and technical specifications. American Academy of Sleep Medicine. 2007

[5]. Hobson JA. A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects. Electroencephalography and Clinical Neurophysiology. 1969; 26(6):644.

[6]. Redmond SJ, Heneghan C. Cardiorespiratory-based sleep staging in subjects with obstructive sleep apnea. IEEE Trans Biomedical Engineering. 2006; 53:485–496.

[7]. Alickovic E, Subasi A. Ensemble SVM method for automatic sleep stage classification. IEEE Trans on Instrumentation and Measurement. 2018; 67(6):1258–1265.

[8]. Phan H, et al. Joint classification and prediction CNN framework for automatic sleep stage classification. IEEE Trans Biomedical Engineering (TBME). 2018

[9]. Supratak A, et al. DeepSleepNet: A model for automatic sleep stage scoring based on raw single-channel EEG. IEEE Trans on Neural Systems and Rehabilitation Engineering. 2017; 25(11): 1998–2008. [PubMed: 28678710]

[10]. Tsinalis O, et al. Automatic sleep stage scoring with single-channel EEG using convolutional neural networks. arXiv:1610.01683. 2016

[11]. Mikkelsen K, De Vos M. Personalizing deep learning models for automatic sleep staging. arXiv: 1801.02645. 2018

[12]. Stephansen JB, et al. Neural network analysis of sleep stages enables efficient diagnosis of narcolepsy. Nature Communications. 2018; 9(1):5229.

[13]. Chambon S, et al. A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series. IEEE Trans on Neural Systems and Rehabilitation Engineering. 2018; 26(4):758–769. [PubMed: 29641380]

[14]. Andreotti, F; , et al. Multichannel sleep stage classification and transfer learning using convolutional neural networks. Proc. EMBC; 2018. 171–174.

[15]. Andreotti, F; Phan, H; De Vos, M. Visualising convolutional neural network decisions in automatic sleep scoring. Proc. Joint Workshop on Artificial Intelligence in Health (AIH); 2018. 70–81.

[16]. Mikkelsen KB, et al. Machine-learning-derived sleep-wake staging from around-the-ear electroencephalogram outperforms manual scoring and actigraphy. Journal of Sleep Research. 2018:e12786. [PubMed: 30421469]

[17]. Looney D, et al. Wearable in-ear encephalography sensor for monitoring sleep. preliminary observations from nap studies. Annals of the American Thoracic Society. 2016; 13(12):32–42.

[18]. Goverdovsky V, Looney D, Mandic PKDP. In-ear EEG from viscoelastic generic earpieces: Robust and unobtrusive 24/7 monitoring. IEEE Sensors Journal. 2016; 16:271–277.

[19]. Kidmose P, et al. A study of evoked potentials from ear-EEG. IEEE Trans Biomedical Engineering. 2013; 60(10):2824–2830.

[20]. Looney D, et al. The in-theear recording concept: User-centered and wearable brain monitoring. IEEE Pulse. 2012; 3(32–42)

[21]. Mikkelsen KB, et al. EEG recorded from the ear: Characterizing the ear-EEG method. Front Neurosci. 2015; 9(438)

[22]. Längkvist M, Karlsson L, Loutfi A. Sleep stage classification using unsupervised feature learning. Advances in Artificial Neural Systems. 2012; 2012:1–9.

[23]. Tsinalis O, Matthews PM, Guo Y. Automatic sleep stage scoring using time-frequency analysis and stacked sparse autoencoders. Annals of Biomedical Engineering. 2016; 44(5):1587–1597. [PubMed: 26464268]

[24]. Phan, H; , et al. Automatic sleep stage classification using single-channel eeg: Learning sequential features with attention-based recurrent neural networks. Proc. EMBC; 2018. 1452–1455.

[25]. Dong H, et al. Mixed neural network approach for temporal sleep stage classification. IEEE Trans on Neural Systems and Rehabilitation Engineering. 2018; 26(2):324–333. [PubMed: 28767373]

Europe PMC Funders Author Manuscripts

[26]. Phan, H; , et al. DNN filter bank improves 1-max pooling CNN for single-channel EEG automatic sleep stage classification. Proc. EMBC; 2018. 453–456.

[27]. Sousa T, et al. A two-step automatic sleep stage classification method with dubious range detection. Computers in Biology and Medicine. 2015; 59:42–53. [PubMed: 25677576]

[28]. Liang, S-F; , et al. A rule-based automatic sleep staging method. Proc. EBMC; 2011. 6067–6070.

[29]. Aboalayon KAI, et al. Sleep stage classification using EEG signal analysis: A comprehensive survey and new investigation. Entropy. 2016; 18(9):272.

[30]. Patanaik A, et al. An end-to-end framework for real-time automatic sleep stage classification. Sleep. 2018; 41(5)

[31]. Bojarski M, et al. End to end learning for self-driving cars. arXiv:1604.07316. 2016

[32]. Collobert R, et al. Natural language processing (almost) from scratch. Journal of Machine Learning Research. 2011:2493–2537.

[33]. Silver D, et al. Mastering the game of Go with deep neural networks and tree search. Nature. 2016; 529(7587):484–489. [PubMed: 26819042]

[34]. Krizhevsky, A; Sutskever, I; Hinton, GE. ImageNet classification with deep convolutional neural networks. Proc. NIPS; 2012. 1097–1105.

[35]. Mnih V, et al. Human-level control through deep reinforcement learning. Nature. 2015; 518(7540):529–533. [PubMed: 25719670]

[36]. Levine S, et al. End-to-end training of deep visuomotor policies. Journal of Machine Learning Research. 2016; 17:1–40.

[37]. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computing. 1997; 9(8):1735–1780.

[38]. Cho, K; , et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. Proc. EMNLP; 2014. 1724–1734.

[39]. O'Reilly C, et al. Montreal archive of sleep studies: An open-access resource for instrument benchmarking & exploratory research. Journal of Sleep Research. 2014:628–635. [PubMed: 24909981]

[40]. Imtiaz, SA; Rodriguez-Villegas, E. Recommendations for performance assessment of automatic sleep staging algorithms. Proc. EMBC; 2014. 5044–5047.

[41]. Imtiaz, SA; Rodriguez-Villegas, E. An open-source toolbox for standardized use of PhysioNet Sleep EDF Expanded Database. Proc. EMBC; 2015. 6014–6017.

[42]. Lajnef T, et al. Learning machines and sleeping brains: Automatic sleep stage classification using decision-tree multi-class support vector machines. Journal of Neuroscience Methods. 2015; 250:94–105. [PubMed: 25629798]

[43]. Huang CS, et al. Knowledge-based identification of sleep stages based on two forehead electroencephalogram channels. Frontiers in Neuroscience. 2014; 8:263. [PubMed: 25237291]

[44]. Yu H, et al. DNN filter bank cepstral coefficients for spoofing detection. IEEE Access. 2017; 5:4779–4787.

[45]. Koch, P; , et al. Recurrent neural network based early prediction of future hand movements. Proc. EMBC; 2018. 4710–4713.

[46]. Koch, P; , et al. Recurrent neural networks with weighting loss for early prediction of hand movements. Proc. EUSIPCO; 2018. 1152–1156.

[47]. Phan H, et al. Improved audio scene classification based on label-tree embeddings and convolutional neural networks. IEEE/ACM Trans on Audio, Speech, and Language Processing. 2017; 25(6):1278–1290.

[48]. Luong, T; Pham, H; Manning, CD. Effective approaches to attention-based neural machine translation. Proc. EMNLP; 2015. 1412–1421.

[49]. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473. 2015

[50]. Chung J, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555. 2014

[51]. Abadi M, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467. 2016

[52]. Srivastava N, et al. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research (JMLR). 2014; 15:1929–1958.

[53]. Cooijmans T, et al. Recurrent batch normalization. arXiv:1603.09025. 2016

[54]. Kingma, DP; Ba, JL. Adam: a method for stochastic optimization. Proc. ICLR; 2015.

[55]. Phan, H; , et al. Audio scene classification with deep recurrent neural networks. Proc. INTERSPEECH; 2017. 3043–3047.

[56]. American Academy of Sleep Medicine. International Classification of Sleep Disorders – Third Edition (ICSD-3). 2014.

[57]. Cooray N, et al. Detection of REM sleep behaviour disorder by automated polysomnography analysis. arXiv:1811.04662. 2018

Europe PMC Funders Author Manuscripts

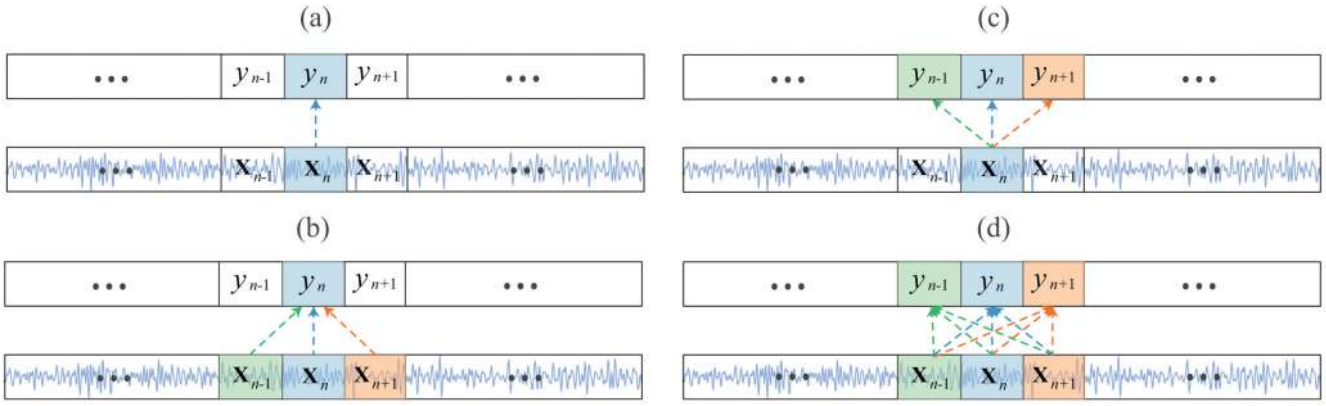

Fig. 1.

Illustration of the classification schemes used for automatic sleep staging. (a) one-to-one, (b) many-to-one, (c) one-to-many, and (d) the proposed many-to-many.

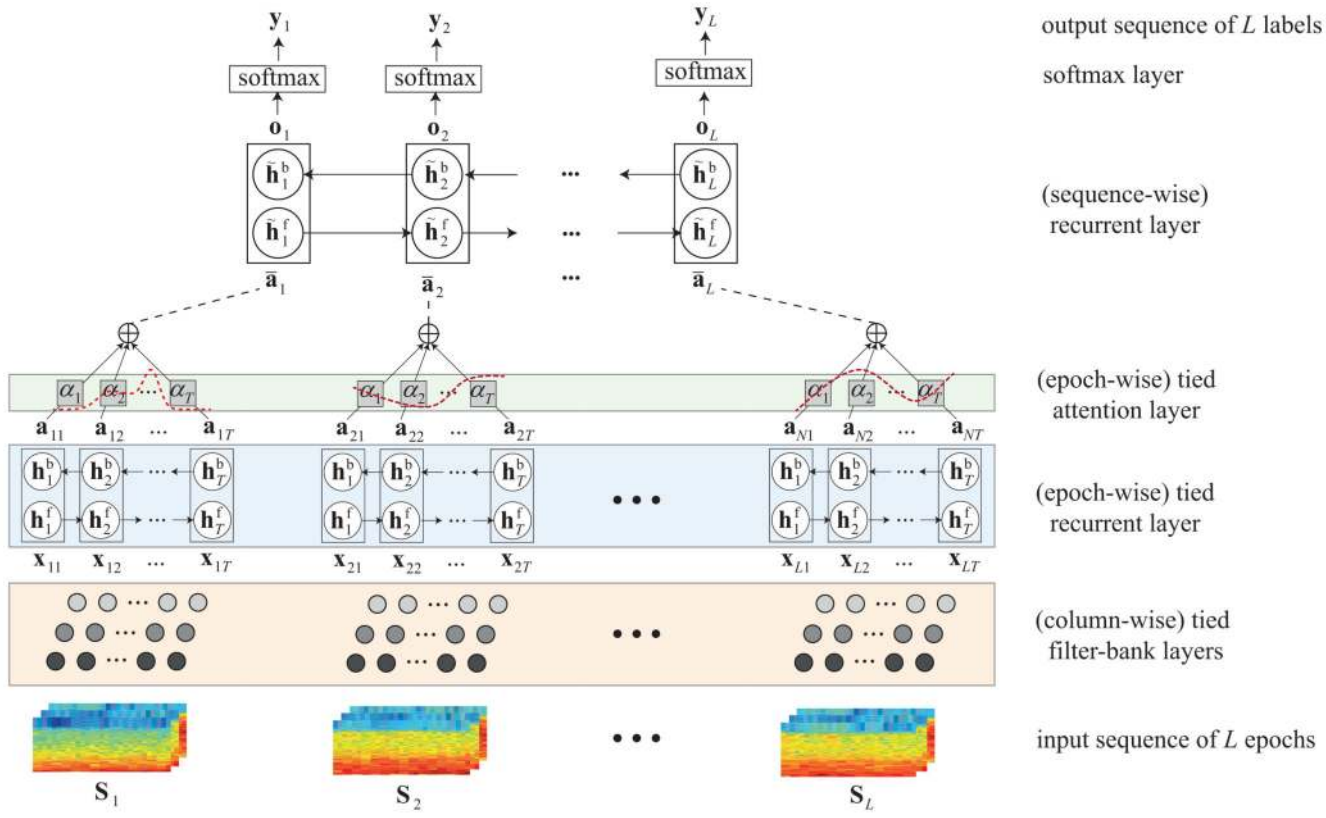

Fig. 2.

Illustration of SeqSleepNet, the proposed end-to-end hierarchical RNN for sequence-to-sequence sleep staging.

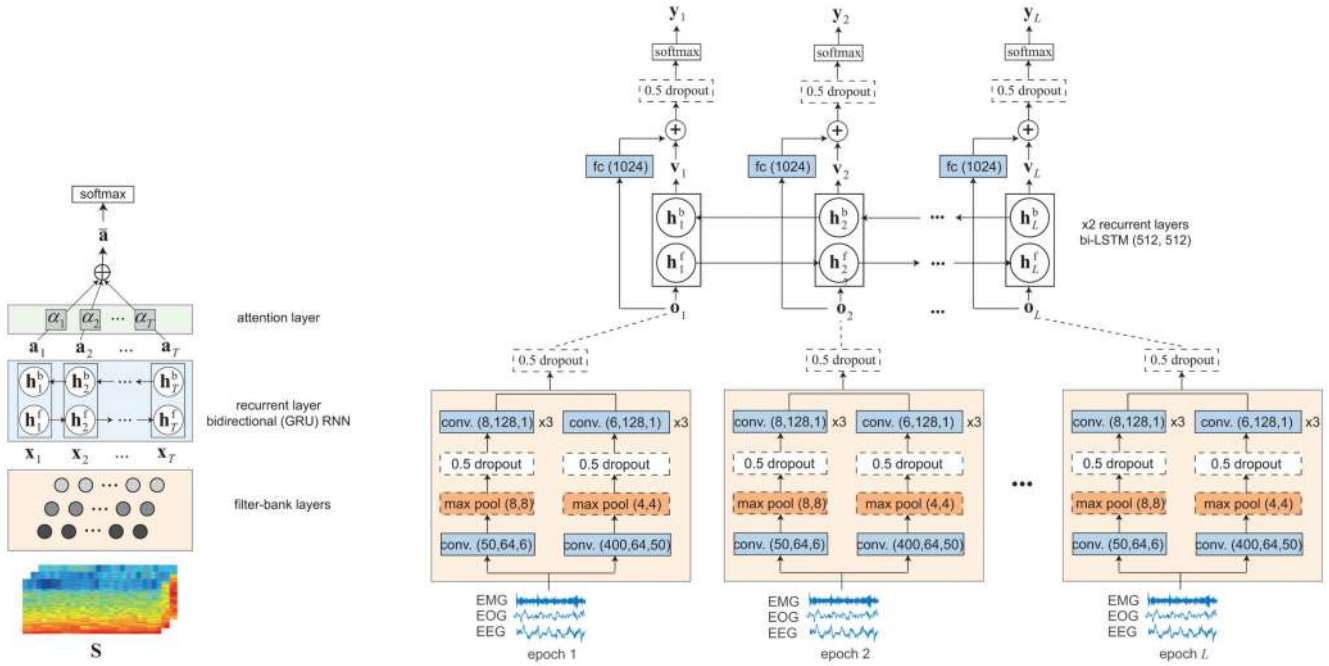

Fig. 3.

Illustration of the developed baselines. In (b), *conv. (n,w,s)* denotes a convolutional layer with *n* 1-D filters of size *w* and stride *s*. *max pool. (w,s)* denotes a 1-D max pooling layer with kernel size *w* and stride *s*. *fc (n)* represents a fully connected layer with *n* hidden units. Finally, *bi-LSTM (n,m)* represents a bidirectional LSTM cell with size of its forward and backward hidden state vectors of *n* and *m*, respectively. Further details of these parameters can be found in [9].

Europe PMC Funders Author Manuscripts

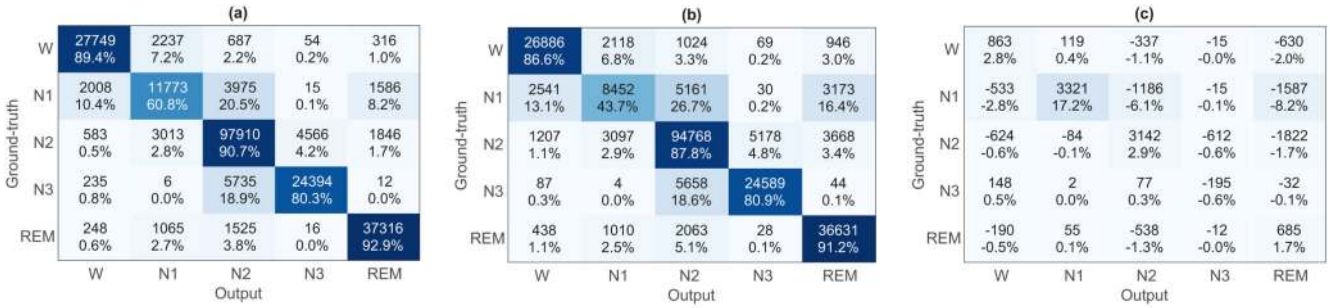

Fig. 4.

(a) The confusion matrix of SeqSleepNet-20 (C_1), (b) the confusion matrix of the E2E-ARNN baseline (C_2), and (c) the difference of two confusion matrices $C_1 - C_2$.

Europe PMC Funders Author Manuscripts

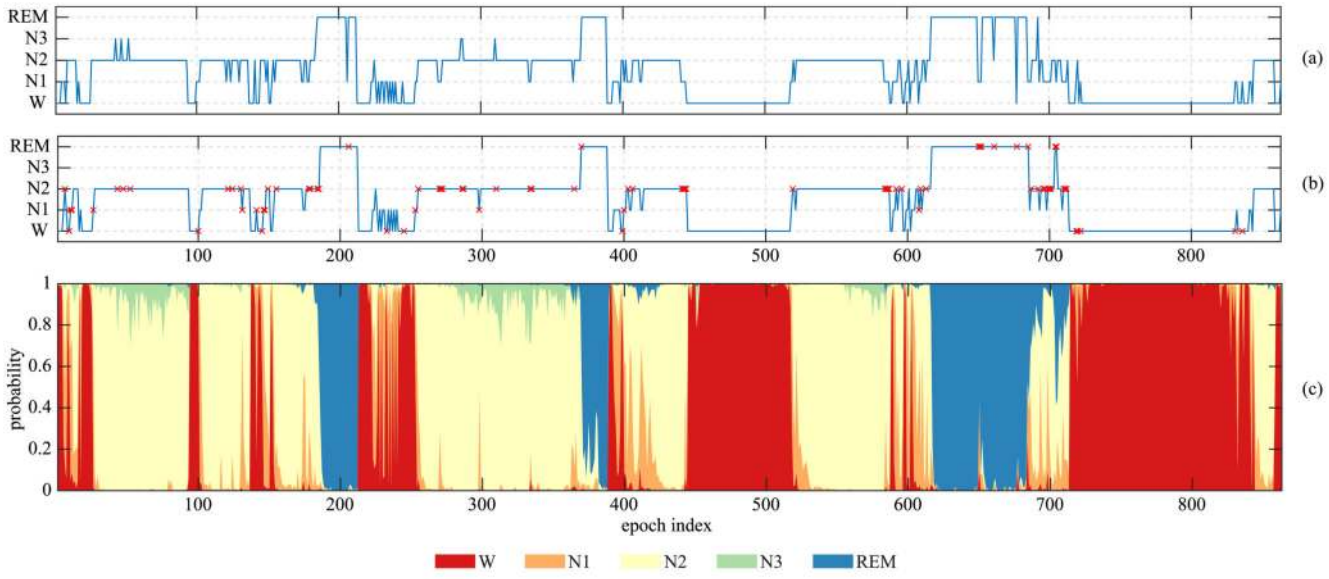

Fig. 5.

Output hypnogram (a) produced by the proposed SeqSleepNet ($L = 20$) for subject 22 of the MASS dataset compared to the ground-truth (b). The errors are marked by the × symbol. The posterior probability distribution over different sleep stages is shown in (c).

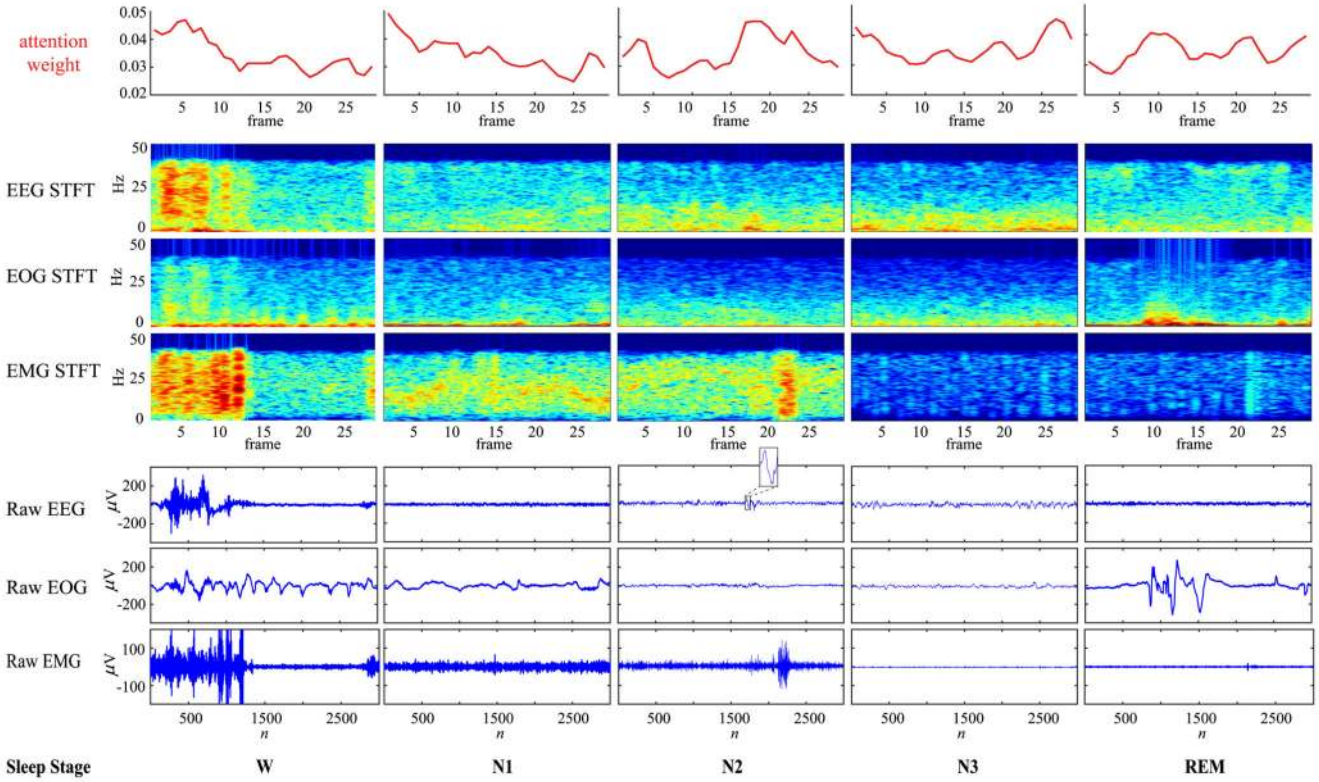

Fig. 6.

Attention weight learned by SeqSleepNet ($L = 20$) for specific epochs of different sleep stages. Note that we generated the spectrograms with finer temporal resolution (2-second window with 90% overlap) for visualization purpose.

Europe PMC Funders Author Manuscripts

Europe PMC Funders Author Manuscripts

Table I

Parameters of the proposed network.

Parameter	Value
Sequence length L	{10, 20, 30}
Number of filters M	32
Size of hidden state vector	64
Size of the attention weights	64
Dropout rate	0.25
Regularization parameter λ	10^{-3}

Europe PMC Funders Author Manuscripts

Table II

Performance obtained by the proposed SeqSleepNet, the developed baselines, and existing works on the MASS dataset. We mark the proposed SeqSleepNet in bold, the developed baselines in italic, and existing works in normal font. SeqSleepNet-L indicates a SeqSleepNet with sequence length of L, a similar notation is used for E2E-DeepSleepNet baseline.

		Method	Feature type	Num. of subjects	Overall metrics					Class-wise sensitivity					Class-wise selectivity				
					Acc.	κ	MF1	Sens.	Spec.	W	N1	N2	N3	REM	W	N1	N2	N3	REM
Multi-output Systems	**SeqSleepNet-30**	ARNN + RNN	learned	200	**87.1**	**0.815**	**83.3**	**82.7**	**96.2**	89.0	**59.7**	**90.9**	80.2	**93.5**	**90.7**	**65.1**	88.9	**84.2**	90.7
	SeqSleepNet-20	ARNN + RNN	learned	200	**87.0**	**0.815**	**83.3**	**82.8**	**96.3**	**89.4**	**60.8**	**90.7**	80.3	**92.9**	**90.0**	**65.1**	**89.1**	**84.0**	90.8
	SeqSleepNet-10	ARNN + RNN	learned	200	**87.0**	**0.814**	**83.2**	**82.4**	**96.2**	88.6	**59.9**	**91.2**	79.4	**93.0**	**91.3**	**64.9**	88.6	**85.1**	90.2
	E2E-DeepSleepNet-30	CNN + RNN	learned	200	86.4	0.805	82.2	81.8	96.1	89.2	55.8	90.5	83.1	90.3	88.8	62.6	88.8	82.0	91.1
	E2E-DeepSleepNet-20	CNN + RNN	learned	200	86.2	0.804	82.2	82.0	96.1	88.4	57.0	89.9	**84.1**	90.4	89.0	62.1	89.0	81.1	**91.2**
	E2E-DeepSleepNet-10	CNN + RNN	learned	200	86.3	0.804	82.0	81.6	96.1	88.4	55.6	90.3	83.4	90.6	88.8	62.0	89.0	82.3	90.2
	M-E2E-ARNN	ARNN	learned	200	83.8	0.767	77.7	77.0	95.3	85.0	37.4	89.2	79.2	94.2	86.5	61.4	86.5	82.6	81.9
	Multitask 1-max CNN [8]	CNN	learned	200	83.6	0.766	77.9	77.4	95.3	84.6	41.1	88.5	79.7	93.3	86.3	55.2	86.9	83.0	83.3
	DeepSleepNet2 [9]	CNN + RNN	learned	62 (SS3)	86.2	0.800	81.7	-	-	-	-	-	-	-	-	-	-	-	-
	Dong *et al.* [25]	DNN + RNN	learned	62 (SS3)	85.9	-	80.5	-	-	-	-	-	-	-	-	-	-	-	-
Single-output Systems	*E2E-ARNN*	ARNN	learned	200	83.6	0.766	78.4	78.0	95.3	86.6	43.7	87.8	80.9	91.2	86.3	57.6	87.2	82.3	82.4
	1-max CNN [8]	CNN	learned	200	82.7	0.754	77.6	77.8	95.1	84.8	46.8	86.4	82.0	88.6	86.2	49.8	87.4	80.2	84.2
	Chambon *et al.* [13]	CNN	learned	200	79.9	0.726	76.7	80.0	95.0	81.1	64.2	76.2	89.6	89.0	86.7	41.0	92.4	73.1	82.6
	DeepSleepNet1 [9]	CNN (only)	learned	200	80.7	0.725	75.8	75.5	94.5	80.0	51.9	85.5	69.0	91.1	87.5	46.2	85.3	84.9	79.7
	Tsinalis *et al.* [10]	CNN	learned	200	77.9	0.680	70.4	69.4	93.5	82.3	30.5	86.8	61.7	85.8	77.5	44.7	80.6	80.0	80.0
	Chambon *et al.* [13]	CNN	learned	61 (SS3)	83.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	DeepSleepNet1 [9]	CNN (only)	learned	62 (SS3)	81.5	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Dong *et al.* [25]	DNN (only)	learned	62 (SS3)	81.4	-	77.2	-	-	-	-	-	-	-	-	-	-	-	-
	Dong *et al.* [25]	RF	hand-crafted	62 (SS3)	81.7	-	72.4	-	-	-	-	-	-	-	-	-	-	-	-
	Dong *et al.* [25]	SVM	hand-crafted	62 (SS3)	79.7	-	75.0	-	-	-	-	-	-	-	-	-	-	-	-

Europe PMC Funders Author Manuscripts

Table III

Influence of SeqSleepNet's recurrent depth on the overall accuracy.

Recurrent depth	Sequence length		
	$L = 10$	$L = 20$	$L = 30$
1	87.0	87.0	87.1
2	86.8	87.0	87.1