# XSleepNet: Multi-View Sequential Model for Automatic Sleep Staging

Huy Phan, Oliver Y. Chén, Minh C. Tran, Philipp Koch,
Alfred Mertins, and Maarten De Vos

**Abstract**—Automating sleep staging is vital to scale up sleep assessment and diagnosis to serve millions experiencing sleep deprivation and disorders and enable longitudinal sleep monitoring in home environments. Learning from raw polysomnography signals and their derived time-frequency image representations has been prevalent. However, learning from multi-view inputs (e.g., both the raw signals and the time-frequency images) for sleep staging is difficult and not well understood. This work proposes a sequence-to-sequence sleep staging model, XSleepNet, that is capable of learning a joint representation from both raw signals and time-frequency images. Since different views may generalize or overfit at different rates, the proposed network is trained such that the learning pace on each view is adapted based on their generalization/overfitting behavior. In simple terms, the learning on a particular view is speeded up when it is generalizing well and slowed down when it is overfitting. View-specific generalization/overfitting measures are computed on-the-fly during the training course and used to derive weights to blend the gradients from different views. As a result, the network is able to retain the representation power of different views in the joint features which represent the underlying distribution better than those learned by each individual view alone. Furthermore, the XSleepNet architecture is principally designed to gain robustness to the amount of training data and to increase the complementarity between the input views. Experimental results on five databases of different sizes show that XSleepNet consistently outperforms the single-view baselines and the multi-view baseline with a simple fusion strategy. Finally, XSleepNet also outperforms prior sleep staging methods and improves previous state-of-the-art results on the experimental databases.

**Index Terms**—Automatic sleep staging, deep neural network, multi-view learning, gradient blending, sequence-to-sequence, end-to-end.

✦

## 1 INTRODUCTION

Anyone who has experienced a sleepless night would acknowledge the importance of sleep in maintaining one's mental and physical health [1], [2]. Unfortunately, sleep deprivation and disorders are prevalent, affecting millions of people worldwide and imposing serious public health issues [3]. For example, 50 to 70 millions of Americans suffer from a chronic sleep or wakefulness disorder, such as insomnia, narcolepsy, restless legs syndrome, and sleep apnea [4]. Additionally, medical errors due to sleep deprivation have caused 100 thousand deaths in US hospitals. Consequently, there is an increasing demand for accurate sleep assessment, diagnosis, and longitudinal monitoring in home environments [5], [6]. In order to address these challenges, it is imperative to employ automated sleep scoring since labor-intensive and time-consuming manual scoring becomes difficult to handle large-scale sleep data.

- H. Phan is with the School of Electronic Engineering, Queen Mary University of London, London, UK, E1 4NS.
  E-mail: h.phan@qmul.ac.uk
- O. Y. Chén is with the Department of Engineering Science, University of Oxford, Oxford, UK, OX1 3PJ.
- M. C. Tran is with Nuffield Division of Anaesthetics, University of Oxford, Oxford, UK, OX3 9DU.
- P. Koch and A. Mertins are with the Institute for Signal Processing, University of Lübeck, Lübeck, Germany, 23562.
- M. De Vos is with the Department of Engineering and with the Department of Development and Regeneration, KU Leuven, Leuven, Belgium, 3001.

1. Source code will be available at http://github.com/pquochuy/xsleepnet.

Consider a task to score an overnight polysomnography (PSG) recording. It takes a sleep expert about two hours to complete the task manually [7]; in contrast, a machine can complete it in a few seconds.

The sleep research community is witnessing an unprecedented progress in automatic sleep staging. Machine's performance is approaching sleep experts' [8], [9]. This is due, in part, to the ever-growing annotated sleep databases. Using large-scale data, novel sleep scoring methods can be developed and tested under powerful deep learning paradigms [9], [10], [11], [12], [13]. Since the earlier attempts [14], [15], deep learning for automatic sleep staging has evolved rapidly in both designing targeted modelling methodologies and building effective network architectures. The standard one-to-one [16], [17] and many-to-one [15], [18] methods are beginning to be replaced with one-to-many (i.e., multitasking) [19] and many-to-many (i.e., sequence-to-sequence) frameworks [8], [20] which better represent the sequential nature of sleep data. Concerning network architectures, the vanilla ones, such as Deep Belief Networks (DBNs) [14], Auto-encoders [21], Deep Neural Networks (DNNs) [22], Convolutional Neural Networks (CNNs) [15], [16], [18], [19], [23], [24], and Recurrent Neural Networks (RNNs) [17], are being surpassed by more complex, task-specific architectures, such as DNN+RNN [22], CNN+RNN [20], and hierarchical RNN [8].

Existing work on automatic sleep staging can be categorized based on the types of signal input of the network. There are two main categories: the first directly processes

1-dimensional raw signals [15], [18], [20], [22], [24], [25], [26] and the second ingests 2-dimensional time-frequency images as inputs [8], [9], [16], [17]. A time-frequency image is usually derived from a raw signal via some transformations, for example, short-time Fourier transform (STFT). It is, in general, considered as a higher-level representation of the raw signal. However, one cannot conclude that the raw input is better than the time-frequency one as the performance of an automatic sleep staging system depends on many other factors, such as the amount of training data, the network architecture, etc. Rather, they should be considered as two different views regarding the same underlying data distribution. Used together, they should complement each other and improve performance of the task at hand than used separately. Indeed, prior works [11], [12], [13] have attempted to combine both raw signals and time-frequency images in the same network to tackle automatic sleep staging. Such a network is designed to have a subnet dedicated to an input type. The learned features from different network subnets are then combined (for example, via concatenation) to form joint features on which classification are made.

In general, learning representations that capture information from multiple views should benefit recognition performance [27], [28]. Confusingly, combining multiple input types in a deep network often results in a performance drop rather than an improvement, as we will show in our experiments. This observation has not been well understood. In this work, we will demonstrate that using a simple strategy like concatenation (as in [11], [12], [13]) to learn from multi-view input is suboptimal. We will also illustrate why a multi-view network often results in worse performance than the best single-view counterpart. To address this issue, we will introduce a sequence-to-sequence network, XSleepNet, that can learn joint features from both raw and time-frequency input effectively. During training, the network oversees overfitting/generalization behavior on the input views and uses this information to adapt their contributions into the joint-feature learning via gradient blending. Simply put, learning on the view that is generalizing well will be encouraged while learning on the view that is overfitting will be discouraged. In addition, we layout the principles that guides the design of XSleepNet to achieve robustness (to the amount of training data) and complementarity (i.e. how the two input views complement one another). To evaluate the efficacy of the model, we conduct experiments on five databases with different sizes and show that XSleepNet outperforms both three strong baselines and existing works on these databases. It is most likely that XSleepNet is able to consolidate the representation power of the input views to produce the joint features which better represent the underlying data distribution, and therefore, results in higher performance than using single views alone or the simple concatenation fusion strategy.

The rest of the article is organized as follows. We outline the principles guiding the network design in Section 2. We describe the network architecture and its multi-view joint learning mechanism in Section 3. Details about the experiments will be presented in Section 4, followed by a discussion in Section 5. We conclude the article in Section 6.

## 2 DESIGN PRINCIPLES

As a multi-view model, XSleepNet is composed of two network streams: one for the raw signal and the other for the time-frequency image. The following design principles aim to introduce robustness (to the amount of training data) and complementarity (i.e. how the two input views complement one another) into the network while maintaining its flexibility to learn from multiple views effectively.

*Principle 1 (Robustness): The raw-data network stream is large while the time-frequency one is compact in terms of model size.* Specifically, the raw-data network has $5.6 \times 10^6$ parameters in total, about 35 times more than $1.6 \times 10^5$ parameters in the time-frequency one. In general, the footprint of a deep network is proportional to its modelling capacity and should be devised depending on the amount of available training data. The rule of thumb is to increase the network capacity when the training data is large, and decrease it otherwise. However, to do so is not trivial, especially for some clinicians, who may not be technology-savvy. With two network streams of varying modelling capacity, XSleepNet is robust in terms of performance regardless of the amount of training data. When the training data is small, the higher-capacity stream may overfit, but the lower-capacity one generalizes well. When the training data is large, the lower-capacity may underfit, but the higher-capacity stream generalizes well. Combining the two streams results in a balanced, and generalizable model. This is possibly owing to the generalization- and overfitting-aware training procedure of the network (see *Principle 3*).

*Principle 2 (Complementarity): The two network streams have diverging architectures.* Theoretically, for a joint model to be effective, each individual model should be diversified [29]. Practically, there is empirical evidence suggesting that CNNs with raw signals and RNNs with time-frequency images are complementary to each other on their sleep staging outputs. For example, the former favors N3, and the latter works better for N1 and REM on MASS database [8], [30]. We, therefore, design XSleepNet such that the raw stream is based on a CNN and the time-frequency stream is based on an RNN to extract epoch-wise features. Even when an RNN is required for inter-epoch sequential modelling, different types of RNN cells are used in the two network streams (see more details in the next section) to ensure the diversification.

*Principle 3 (Generalization- and overfitting-aware training): The multi-view network is trained such that learning on the network stream that is generalizing well is accelerated while the overfitting one is discouraged.* This principle is a requirement without which the multi-view network would fail to produce better representation than training either of the single-view network streams separately. In literature, a network with multiple input types typically combines the features learned from the constituent streams (e.g., via concatenation [11], [12], [13]) to form the joint representation which then serves the classification purpose. This is illustrated in Fig. 1 (a). As a result, there is not a viable way to regulate the learning pace of the streams individually. This would not be a problem if the network streams generalize and overfit at the same time. However, this is not often the case, as illustrated via their validation losses in Fig. 1 (b). In turn,
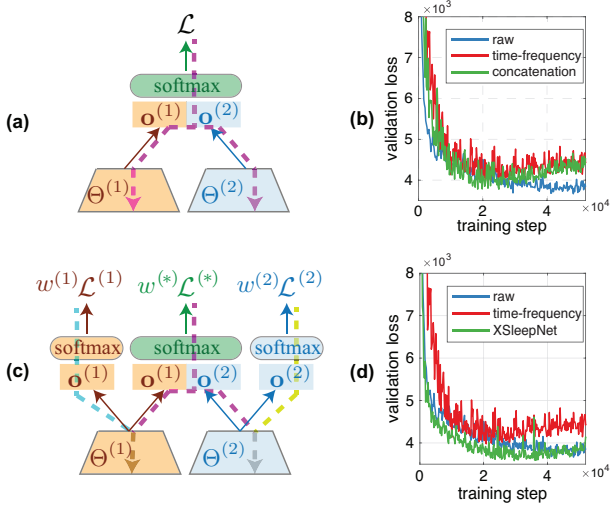
Fig. 1: Illustration of multi-view learning with simple concatenation and XSleepNet. Joint multi-view learning with simple concatenation (a) and its resulting validation loss in comparison with that of single-view training (b). Joint multi-view learning with XSleepNet (c) and its resulting validation loss in comparison with that of single-view training. In (a) and (c), the dashed lines represent the gradient flows. $\boldsymbol{\Theta}$, $\mathbf{o}$, and $\mathcal{L}$ denote a network stream corresponding to one input view, a learned feature vector, and a loss value, respectively. The superscripts, i.e. (1) and (2), indicate the input view specifically. In addition, in (c), $w$ denotes a weight and the superscript (*) indicates the joint network branch.

the validation loss of the simple combination appears to be averaged out as illustrated in the figure, suggesting worse generalization than the best single-view network stream.

In order to regulate the learning pace of the network streams, it is necessary to gain access to their gradient flows. In XSleepNet, in addition to the joint classification branch, two additional branches are added. Different from the joint classification branch, these two newly introduced branches operate on the stream-wise features as illustrated in Fig. 1 (c). By monitoring the generalization/overfitting behavior (see more details in Section 3.2) of the classification branches, we are able to weight their gradient flows so that the one that generalizes well is awarded a large weight and the one that overfits is given a small weight. By doing so, we blend the gradients according to the generalization and overfitting behavior of the classification branches and individualize the learning pace of the network streams. Unlike the simple concatenation (see Fig. 1 (b)), with this adaptive gradient blending approach, XSleepNet results in a better joint representation of the underlying data distribution than that of the single-view networks, as evidenced by its validation loss in Fig. 1 (d).

## 3 XSLEEPNET

### 3.1 Architecture

Given a training set $\{\mathcal{S}_n\}_{n=1}^{N}$ of size $N$ where $\mathcal{S} = \left( \left\{ (\mathbf{S}_1^{(1)}, \mathbf{S}_1^{(2)}), \mathbf{y}_1 \right\}, \dots, \left\{ (\mathbf{S}_L^{(1)}, \mathbf{S}_L^{(2)}), \mathbf{y}_L \right\} \right)$ is a sequence of $L$ sleep epochs. $\mathbf{S}_l^{(1)}$ and $\mathbf{S}_l^{(2)}$ represent two different views (i.e., inputs of two different types) and

$\mathbf{y}_l \in \{0,1\}^Y$ denotes the one-hot encoding label of the $l$-th epoch, for $1 \leq l \leq L$. $Y = 5$ since we consider the classification of 5 sleep stages in this work. $\mathbf{S}_l^{(1)} \in \mathbb{R}^{3000 \times C}$ is a $C$-channel 30-second raw signal sampled at 100 Hz and $\mathbf{S}_l^{(2)} \in \mathbb{R}^{F \times T \times C}$ is the time-frequency representation, with $F = 129$ frequency bins and $T = 29$ time points (see more details in Section 4). Here, $C$ takes a value of $\{1, 2, 3\}$ depending on the used channel combination (i.e., EEG, EEG·EOG, or EEG·EOG·EMG).

The architecture of XSleepNet is illustrated in Fig. 2. The network stream in orange handles the raw signal and the one in blue deals with the time-frequency image. Long-term (i.e., inter-epoch) sequential modelling is at the heart of sequence-to-sequence sleep staging models, including XSleepNet. In light of this, we employ bidirectional RNNs for this purpose as in [8], [20]. The raw and time-frequency inputs are therefore encoded into two sequences of output vectors, respectively:

$$(\mathbf{o}_1^{(1)}, \dots, \mathbf{o}_L^{(1)}) = GRU(\mathcal{F}_1(\mathbf{S}_1^{(1)}), \dots, \mathcal{F}_1(\mathbf{S}_L^{(1)})), \qquad (1)$$

$$(\mathbf{o}_1^{(2)}, \dots, \mathbf{o}_L^{(2)}) = LSTM(\mathcal{F}_2(\mathbf{S}_1^{(2)}), \dots, \mathcal{F}_2(\mathbf{S}_L^{(2)})). \qquad (2)$$

With respect to the design Principle 2, the bidirectional RNN in (1) is realized by Gated Recurrent Unit (GRU) cells [31] and the one in (2) is realized by Long Short-Term Memory (LSTM) cells [32] coupled with recurrent batch normalization [33]. In (1) and (2), $\mathbf{o}_l^{(1)} \in \mathbb{R}^{2H_1}$ and $\mathbf{o}_l^{(2)} \in \mathbb{R}^{2H_2}$, for $1 \leq l \leq L$, where $H_1$ and $H_2$ are the sizes of the GRU and LSTM cells' hidden state vectors, respectively. $\mathcal{F}_1$ and $\mathcal{F}_2$ denote the subnetworks that play the role of a feature map to transform an input epoch into a feature vector of high-level representation. They are separately tailored for raw and time-frequency inputs with respect to the design Principle 2.

**Feature map $\mathcal{F}_1$:** In order to transform a raw signal $\mathbf{S}^{(1)}$ into a high-level feature $\mathbf{x}^{(1)}$, the feature map $\mathcal{F}_1 : \mathbf{S}^{(1)} \mapsto \mathbf{x}^{(1)}$ is realized by a *fully* convolutional neural network (FCNN) [34]. Without confusion, the subscript $l$ is omitted here for simplicity. The network consists of 9 strided one-dimensional convolutional layers [34] with a common filter width of 31 and stride length of 2. The convolutional layers are designed to have their numbers of filter increased according to the network's depth, taking values of $16, 16, 32, 32, 64, 64, 128, 128$, and $256$, to compensate for the gradually smaller induced feature maps. Given the input $\mathbf{S}^{(1)}$ of size $3000 \times C$, the CNN results in feature map sizes of $1500 \times 16$, $750 \times 16$, $325 \times 32$, $163 \times 32$, $82 \times 64$, $41 \times 64$, $21 \times 128$, $11 \times 128$, and $6 \times 256$ after the 9 convolutional layers, respectively. In addition, each convolutional layer is followed by parametric rectified linear units (PReLUs) [35]. The output of the last convolutional layer is flattened to form the induced epoch-wise feature vector $\mathbf{x}^{(1)} = \mathcal{F}_1\left(\mathbf{S}^{(1)}\right) \in \mathbb{R}^{1536}$.

**Feature map $\mathcal{F}_2$:** Different from $\mathcal{F}_1$, the feature map $\mathcal{F}_2 : \mathbf{S}^{(2)} \mapsto \mathbf{x}^{(2)}$ relies on the attention-based RNN coupled with learnable *filterbank layers* to map a time-frequency input $\mathbf{S}^{(2)}$ to a high-level feature vector $\mathbf{x}^{(2)}$. Again, the subscript $l$ is omitted for simplicity. First, each channel of $\mathbf{S}^{(2)}$ is preprocessed by a learnable filterbank layer of $D$ filters as in [8], [16] so that its spectral dimension is
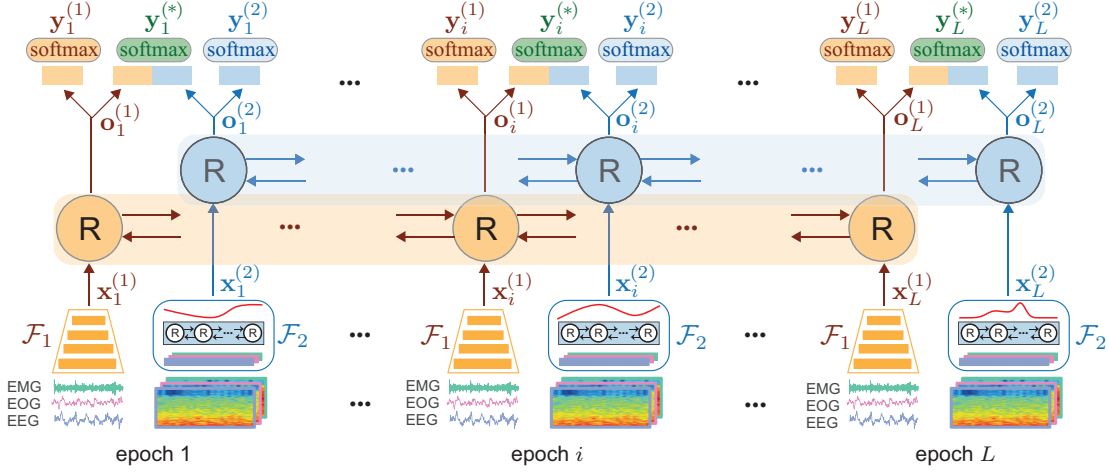
Fig. 2: The architecture of XSleepNet. The network streams orange and blue correspond to the raw and time-frequency image inputs, respectively.

smoothed and reduced from $F$ to $D$ frequency bins, resulting in $\mathbf{S}'^{(2)} \in \mathbb{R}^{T \times D \times C}$. Afterwards, $\mathbf{S}'^{(2)}$ is interpreted as a sequence of $T$ vectors $(\mathbf{s}'_1, \ldots, \mathbf{s}'_T)$, where each $\mathbf{s}'_t \in \mathbb{R}^{DC}$, for $1 \le t \le T$, and encoded by a bidirectional RNN into a sequence of vectors $(\mathbf{z}_1, \ldots, \mathbf{z}_T)$:

$$(\mathbf{z}_1, \ldots, \mathbf{z}_T) = LSTM(\mathbf{s}'_1, \ldots, \mathbf{s}'_T). \tag{3}$$

Here the bidirectional RNN is realized by LSTM cells [32] coupled with recurrent batch normalization [33] and $\mathbf{z}_t \in \mathbb{R}^{2H}$ with $H$ equal to the size of the LSTM cells' hidden state vectors. Note that this bidirectional RNN here is for short-term (i.e. intra-epoch) sequential modelling and should not be confused with the one for inter-epoch sequential modelling in (2). The induced epoch-wise feature vector $\mathbf{x}^{(2)}$ is eventually obtained via a weighted combination of the vectors $\mathbf{z}_1, \ldots, \mathbf{z}_T$:

$$\mathbf{x}^{(2)} = \sum_{t=1}^{T} \alpha_t \mathbf{z}_t. \tag{4}$$

In (4), $\alpha_1, \ldots, \alpha_T$ are attention weights learned by an attention layer as in [8], [17]:

$$\alpha_t = \frac{\exp(\mathbf{a}_t^\mathsf{T} \mathbf{a}_e)}{\sum_{t=1}^{T} \exp(\mathbf{a}_t^\mathsf{T} \mathbf{a}_e)}, \tag{5}$$

$$\mathbf{a}_t = \tanh(\mathbf{W}_a \mathbf{z}_t + \mathbf{b}_a), \tag{6}$$

where $\mathbf{W}_a \in \mathbb{R}^{A \times 2H}$ and $\mathbf{b}_a \in \mathbb{R}^A$ are trainable weight matrix and bias vector, respectively. $\mathbf{a}_e \in \mathbb{R}^A$ is the trainable epoch-level context vector. $A$ is the so-called attention size.

In light of the design Principle 2, it is worth noting that the ways $\mathcal{F}_1$ and $\mathcal{F}_2$ learn to produce features are distinguishable. On the one hand, we conjecture that, due to its reliance on FCNN, $\mathcal{F}_1$ can capture temporal-equivariant patterns from the raw input. That is, such a feature can occur at any time point in the 30-second duration of the raw signal. For example, micro events such as K-complex and sleep spindle [36], [37] frequently appearing in the sleep stage N2 have this characteristic. On the other hand, built upon the bidirectional RNN and the attention mechanism, $\mathcal{F}_2$ can encode the sequential patterns of the spectral columns in its time-frequency image input. This is useful to capture features with a sequential structure, such as the theta wave activity in the sleep stage N1 [36], [37].

Adhering to the design Principle 3, XSleepNet accommodates three softmax branches for classification: the first two are view-specific (i.e., operating on the output vectors of the raw and time-frequency network streams specifically) and the third one for the joint view (i.e., operating on the joint feature vector), as illustrated in Fig. 1. Given the output sequence $(\mathbf{o}_1^{(1)}, \ldots, \mathbf{o}_L^{(1)})$ in (1) and $(\mathbf{o}_1^{(2)}, \ldots, \mathbf{o}_L^{(2)})$ in (2), we obtain three sequences of classification labels $(\hat{\mathbf{y}}_1^{(1)}, \ldots, \hat{\mathbf{y}}_L^{(1)})$, $(\hat{\mathbf{y}}_1^{(2)}, \ldots, \hat{\mathbf{y}}_L^{(2)})$, and $(\hat{\mathbf{y}}_1^{(*)}, \ldots, \hat{\mathbf{y}}_L^{(*)})$ via the three classification layers, where

$$\hat{\mathbf{y}}_l^{(1)} = \mathrm{softmax}(\mathbf{o}_l^{(1)}), \tag{7}$$

$$\hat{\mathbf{y}}_l^{(2)} = \mathrm{softmax}(\mathbf{o}_l^{(2)}), \tag{8}$$

$$\hat{\mathbf{y}}_l^{(*)} = \mathrm{softmax}([\mathbf{o}_l^{(1)} \oplus \mathbf{o}_l^{(2)}]), 1 \le l \le L, \tag{9}$$

and $\oplus$ denotes vector concatenation.

During training, the losses induced by these three classification branches are used to compute the weights for gradient blending (cf. Section 3.2). When the trained network is evaluated on the test data, the joint output $(\hat{\mathbf{y}}_1^{(*)}, \ldots, \hat{\mathbf{y}}_L^{(*)})$ are considered as the final outputs.

## 3.2 Adaptive Gradient Blending for Multi-View Training

Given three classification outputs $(\hat{\mathbf{y}}_1^{(1)}, \ldots, \hat{\mathbf{y}}_L^{(1)})$, $(\hat{\mathbf{y}}_1^{(2)}, \ldots, \hat{\mathbf{y}}_L^{(2)})$, and $(\hat{\mathbf{y}}_1^{(*)}, \ldots, \hat{\mathbf{y}}_L^{(*)})$, the cross-entropy losses induced by the three classification branches are

$$\mathcal{L}^{(k)} = -\frac{1}{L} \sum_{l=1}^{L} \mathbf{y}_l \log(\hat{\mathbf{y}}_l^{(k)}), \tag{10}$$

where $k \in \{1, 2, *\}$. In order to balance the generalization/overfitting rate of the classification branches, we weight the losses adaptively to schedule the learning on the two network streams. Similar to [38], we computed the loss weights using the ratio of generalization and overfitting measures (see Appendix A for a theoretical justification):

$$w^{(k)} = \frac{1}{Z} \frac{G_k}{O_k^2}. \tag{11}$$

In (11), $Z$ is a normalization factor. The generalization measure $G_k$ is defined as the gained information about the target distribution we learn from training. The overfitting
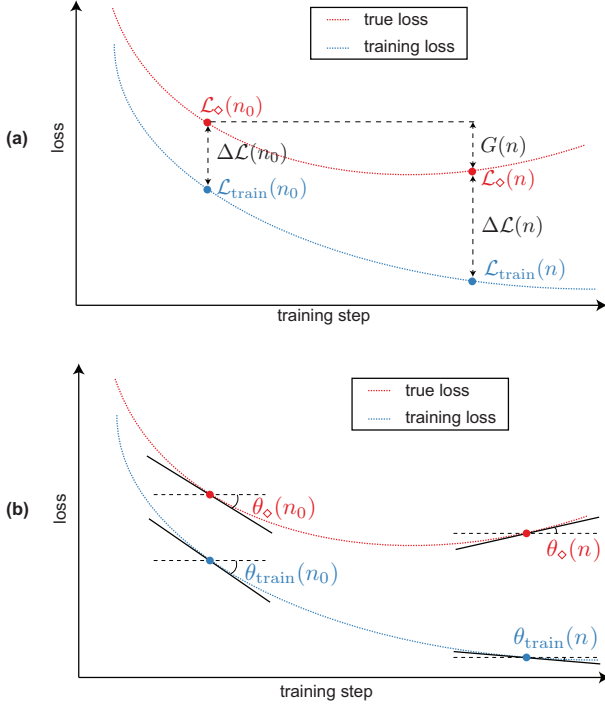
Fig. 3: Theoretical illustration of the interrelation between $\mathcal{L}_{\text{train}}^{(k)}(n)$, $\mathcal{L}_{\diamond}^{(k)}(n)$, $\mathcal{L}_{\text{train}}^{(k)}(n_0)$ and $\mathcal{L}_{\diamond}^{(k)}(n_0)$ in XSleepNet1 (a); and the interrelation between $\theta_{\text{train}}^{(k)}(n)$, $\theta_{\diamond}^{(k)}(n)$, $\theta_{\text{train}}^{(k)}(n_0)$ and $\theta_{\diamond}^{(k)}(n_0)$ in XSleepNet2 (b).

measure $O_k$ is defined as the gap between the gain on the training set and the target distribution. The weighted loss used for training at the training step $n$ is then given by:

$$\mathcal{L}(n) = \sum_{k \in \{1,2,*\}} w^{(k)}(n) \mathcal{L}^{(k)}(n). \tag{12}$$

Using the weighted loss function in (12), the network's learning behavior depends on how $G_k$ and $O_k$ in (11) are approximated. We present two approximation approaches for XSleepNet, resulting in two algorithms, namely *XSleepNet1* and *XSleepNet2*.

**XSleepNet1**: Inspired by [38], we approximate $G_k$ and $O_k$ at the training step $n$ as:

$$G_k(n) \approx \mathcal{L}_{\diamond}^{(k)}(n_0) - \mathcal{L}_{\diamond}^{(k)}(n), \tag{13}$$
$$O_k(n) \approx \Delta\mathcal{L}(n) - \Delta\mathcal{L}(n_0)$$
$$\approx \left[\mathcal{L}_{\diamond}^{(k)}(n) - \mathcal{L}_{\text{train}}^{(k)}(n)\right] - \left[\mathcal{L}_{\diamond}^{(k)}(n_0) - \mathcal{L}_{\text{train}}^{(k)}(n_0)\right], \tag{14}$$

where $\mathcal{L}_{\text{train}}^{(k)}(n)$ and $\mathcal{L}_{\diamond}^{(k)}(n)$ denote the training loss and the true loss at the training step $n$, respectively. $\mathcal{L}_{\text{train}}^{(k)}(n_0)$ and $\mathcal{L}_{\diamond}^{(k)}(n_0)$, where $n_0 < n$, denote the training and true loss references, respectively. The interrelation between $\mathcal{L}_{\text{train}}^{(k)}(n)$, $\mathcal{L}_{\diamond}^{(k)}(n)$, $\mathcal{L}_{\text{train}}^{(k)}(n_0)$ and $\mathcal{L}_{\diamond}^{(k)}(n_0)$ is illustrated in Fig. 3 (a). As in [38], we set $n_0 = 0$ (i.e., right after the network has been initialized with random weights).

**XSleepNet2**: There are at least two potential drawbacks in the approximation approach used in XSleepNet1. First, in theory, the (training and validation) loss curves are smooth (as illustrated in Fig. 3), yet they are typically noisy in practice (cf. Fig. 1) due to minibatch training. As a result, using the spontaneous loss values $\mathcal{L}_{\text{train}}^{(k)}(n)$ and $\mathcal{L}_{\diamond}^{(k)}(n)$

**Algorithm 1** Procedure for computing the loss weight using XSleepNet2's approximation.

1: **procedure** XSLEEPNET2_WEIGHT($L_{\text{train}}, L_{\diamond}, W, n_0$)
2:     **input:** $L_{\text{train}}[1 \ldots n]$: array of training loss values
3:         $L_{\diamond}[1 \ldots n]$: array true loss values
4:         $W$: window size for line fitting
5:     **output:** $w(n)$: weight at the training time $n$
6:         $n_0$: updated reference time step
7:     **if** $n < W$ **then return**
8:     $\tan\theta_{\text{train}}(n) \leftarrow \text{LineFit}(L_{\text{train}}[(n-W)\ldots n])$
9:     $\tan\theta_{\diamond}(n) \leftarrow \text{LineFit}(L_{\diamond}[(n-W)\ldots n])$
10:     $G(n) = \tan\theta_{\diamond}(n) - \tan\theta_{\diamond}(n_0)$
11:     $O(n) = [\tan\theta_{\diamond}(n) - \tan\theta_{\text{train}}(n)] - [\tan\theta_{\diamond}(n_0) - \tan\theta_{\text{train}}(n_0)]$
12:     $w(n) = \frac{1}{Z}\frac{G(n)}{O^2(n)}$
13:     **if** $\tan\theta_{\diamond}(n_0) > \tan\theta_{\diamond}(n)$ **then**
14:         $n_0 = n$    ▷ Time step w.r.t. the true loss's smallest tangent

in the approximation leads to considerably varying loss weights which result in unstable learning behavior and suboptimal performance, particularly when the data size is small. Second, the references $\mathcal{L}_{\text{train}}^{(k)}(0)$ and $\mathcal{L}_{\diamond}^{(k)}(0)$ may vary significantly with different random initializations. This variation further adds up to the approximation suboptimality.

In XSleepNet2, we introduce a novel approximation approach relying on tangents of the losses, which are in turn robustly approximated via line fitting, to overcome these drawbacks. This approach is based on two observations about the loss curves. First, although noisy, a loss curve's overall trend is smooth (i.e., changing slowly). Furthermore, the loss trend at a training step $n$ can be well represented by the direction of its tangent line at $n$. Second, the directions of the loss tangent lines at $n$ are indicative of the network's generalization/overfitting behavior. Let $\theta_{\text{train}}^{(k)}(n)$, and $\theta_{\diamond}^{(k)}(n)$, where $-90° \leq \theta_{\text{train}}^{(k)}(n) \leq 0°$ and $-90° \leq \theta_{\diamond}^{(k)}(n) \leq 90°$, denote the angles made by the tangent lines of the training and true loss curves with the horizontal axis, respectively, as illustrated in Fig 3 (b). The network is generalizing when $-90° \leq \theta_{\diamond}^{(k)}(n) < 0°$ (i.e., negative tangent) and overfitting when $0° < \theta_{\diamond}^{(k)}(n) \leq 90°$ (i.e., positive tangent). In this light, $G_k$ and $O_k$ are approximated as:

$$G_k(n) \approx \tan\theta_{\diamond}^{(k)}(n) - \tan\theta_{\diamond}^{(k)}(n_0), \tag{15}$$
$$O_k(n) \approx \left[\tan\theta_{\diamond}^{(k)}(n) - \tan\theta_{\text{train}}^{(k)}(n)\right]$$
$$- \left[\tan\theta_{\diamond}^{(k)}(n_0) - \tan\theta_{\text{train}}^{(k)}(n_0)\right], \tag{16}$$

where $\tan\theta_{\text{train}}^{(k)}(n)$ and $\tan\theta_{\diamond}^{(k)}(n)$ are the training and true loss tangents at the training step $n$ and $\tan\theta_{\text{train}}^{(k)}(n_0)$ and $\tan\theta_{\diamond}^{(k)}(n_0)$ are their references, respectively. Since a loss tangent is the ratio of the loss change to the increase of the training step (i.e., the slope of the tangent line), the approximations in (15) and (16) are second-order whereas those in (13) and (14) (i.e., in XSleepNet1) are first-order.

The procedure for computing the adaptive loss weight $w^{(k)}$ of the network branch $k$ using XSleepNet2's approximation is described in Algorithm 1. We approximate a loss tangent (i.e., $\tan\theta_{\text{train}}^{(k)}(n)$ and $\tan\theta_{\diamond}^{(k)}(n)$) at the time step $n$ by the slope of the lines fitted to the loss curve segment of length $W$, starting from the time step $n-W$ and proceeding up to $n$. Because of this, the network training starts with a warm-up period of length $W$ during which the weights of

all the network branches are set to be equal. In addition, on each network branch, the time step with respect to the smallest true loss tangent (i.e., the steepest slope of the fitting line) is used as the reference time step $n_0$.

In practice, in both XSleepNet1 and XSleepNet2 the true loss $\mathcal{L}_\diamond^{(k)}(n)$ is unknown. We, therefore, approximate it by the loss evaluated on a held-out validation set, $\mathcal{L}_{\text{valid}}^{(k)}(n)$. To reduce the computational cost of obtaining the training loss $\mathcal{L}_{\text{train}}^{(k)}(n)$ on the entire training set, only a subset randomly sampled from the training set is used for this purpose. Note that, although the second-order approximations in (15) and (16) require more computation than the first-order ones in (13) and (14), the extra cost is negligible compared to the networks' computation. Hence, there is almost no difference between the networks in terms of computational cost.

## 4 EXPERIMENTS

### 4.1 Databases

We employed five publicly available databases in this study: SleepEDF-20, SleepEDF-78, Montreal Archive of Sleep Studies (MASS), Physonet2018, and Sleep Heart Health Study (SHHS). A summary of the databases is shown in Table 1.

**SleepEDF-20:** This is the Sleep Cassette (SC) subset of the Sleep-EDF Expanded dataset [39], [40] (version 2013), consisting of 20 subjects (10 males and 10 females) aged 25-34. Two consecutive day-night PSG recordings were collected for each subject, except for subject 13 who had one night's data lost due to device failure. Each 30-second PSG epoch was manually labelled into one of eight categories {W, N1, N2, N3, N4, REM, MOVEMENT, UNKNOWN} by sleep experts according to the R&K standard [37]. Similar to previous works [15], [16], [17], [19], [20], [21], N3 and N4 stages were considered as N3 collectively and MOVEMENT and UNKNOWN categories were excluded. We adopted the Fpz-Cz EEG and ROC-LOC EOG (i.e., the EOG horizontal) channels in this study. However, we did not experiment with EMG as full EMG recordings are not available.

**SleepEDF-78:** This database is the 2018 version of the Sleep-EDF Expanded dataset [39], [40], consisting of 78 healthy Caucasian subjects aged 25-101. Similar to SleepEDF-20, two consecutive day-night PSG recordings were collected for each subject, except subjects 13, 36, and 52 whose one recording was lost due to device failure. Manual scoring was done by sleep experts according to the R&K standard [37] and each 30-second PSG epoch was labeled as one of eight categories {W, N1, N2, N3, N4, REM, MOVEMENT, UNKNOWN}. N3 and N4 stages were merged into N3 stage. MOVEMENT and UNKNOWN epochs were excluded. We used the Fpz-Cz EEG and ROC-LOC EOG in this study and no experiments were carried out with EMG due to its unavailability.

**MASS:** This database was pooled from different hospital-based sleep laboratories, consisting of whole-night recordings from 200 subjects (97 males and 103 females) aged 18-76. Manual annotation was accomplished by sleep experts according to the AASM standard [36] (SS1 and SS3 subsets) or the R&K standard [37] (SS2, SS4, and SS5 subsets). As in [8], [19], we converted R&K annotations into five sleep stages {W, N1, N2, N3, REM} according to the AASM standard. Epochs with a length of 20 seconds were expanded to 30-second ones by including 5-second segments before and after them. We used C4-A1 EEG, ROC-LOC EOG, and CHIN1-CHIN2 EMG in our experiments.

**Physio2018:** This database was contributed by Massachusetts General Hospital's Computational Clinical Neurophysiology Laboratory. It was used in the 2018 Physionet challenge [40], [41] to detect arousal during sleep. We employed the training set (annotation of the evaluation set was not publicly available) consisting of 944 subjects aged 18-90 in the experiments. Manual scoring was done by sleep experts according to the AASM guideline [36]. C3-A2 EEG, E1-M2 EOG, and CHIN1-CHIN2 EMG were used.

**SHHS:** The SHHS database [42], [43] was collected from multiple centers to study the effect of sleep-disordered breathing on cardiovascular diseases. It has two rounds of PSG records, namely Visit 1 (SHHS-1) and Visit 2 (SHHS-2). The former, consisting of 5,791 subjects aged 39-90, was employed in this work. Manual scoring was completed using the R&K guideline [37]. Similar to other databases annotated with the R&K rule, N3 and N4 stages were merged into N3 stage and MOVEMENT and UNKNOWN epochs were discarded. We adopted C4-A1 EEG, ROC-LOC EOG, and bipolar submental EMG in the experiments.

These databases were adopted in this work to show the robustness of XSleepNet in dealing with datasets of different sizes, which is one of the guiding principles of the network design (see more details in Section 2). For each database, we experimented with single-channel EEG, 2-channel EEG·EOG, and 3-channel EEG·EOG·EMG combinations as inputs where possible. Particularly, only single-channel and 2-channel experiments were conducted on SleepEDF-20 and SleepEDF-78 since they do not have a full EMG channel available in the PSG recordings. All the signals were resampled to 100 Hz.

We conducted experiments following the data splits as summarized in Table 1. These data splits have been commonly used in literature, enabling a direct performance comparison between our system and prior works. As noted in the table, a number of subjects were held out for validation. Particularly, for the SHHS database, we did not leave out a validation set as large as 20% of the subjects as in Sors *et al.* [24]; rather, we left out 100 subjects as we empirically found 100 subjects were sufficient for validation purpose. Furthermore, a random subset was also drawn from the training subjects to compute the loss weights as described in Section 3.2. Of note, different from the validation set, which is not involved in network training, this training subset contributes to the training process as usual.

### 4.2 Parameters

To extract the time-frequency input mentioned in Section 4.2, a signal (i.e., EEG or EOG or EMG) of a 30-second PSG epoch sampled at 100 Hz was divided into two-second windows with 50% overlap, multiplied with a Hamming window, and transformed to the frequency domain using a 256-point Fast Fourier Transform (FFT). The amplitude spectrum was then log-transformed. The time-frequency images extracted from a database were normalized to zero mean and unit standard deviation before training and testing.

The network implementation was based on *Tensorflow* framework [44]. The input sequence length was set to $L=20$

TABLE 1: Summary of the employed databases.

| Database | Size | EEG channel | EOG channel | EMG channel | Scoring method | Experimental setup | Held-out validation set | Random training subset |
|---|---|---|---|---|---|---|---|---|
| SleepEDF-20 | 20 | Fpz-Cz | ROC-LOC | – | R&K | 20-fold CV | 4 subjects | 4 subjects |
| SleepEDF-78 | 79 | Fpz-Cz | ROC-LOC | – | R&K | 10-fold CV | 7 subjects | 7 subjects |
| MASS | 200 | C4-A1/C3-A2 | ROC-LOC | CHIN1-CHIN2 | AASM/R&K | 20-fold CV | 10 subjects | 10 subjects |
| Physio2018 | 994 | C3-A2 | E1-M2 | CHIN1-CHIN2 | AASM | 5-fold CV | 50 subjects | 50 subjects |
| SHHS | 5,791 | C4-A1 | ROC-LOC | Submental EMG | R&K | train/test: 0.7/0.3 | 100 subjects | 100 subjects |

which was proven to be a reasonable value for sequence-to-sequence models [8]. The feature map $\mathcal{F}_2$ was designed with $D = 32$ filters in each filterbank layer, 64 units in its LSTM cells' hidden state vectors, and the attention size $A = 64$ (see Appendix E for an ablation study on the epoch-level attention weights in (5)). The LSTM and GRU cells responsible for inter-epoch sequential modelling had 256 and 64 units in their hidden state vectors, respectively. Throughout the network, dropout [45] was applied during training with a dropout rate of $0.5$ and $0.25$ used for convolutional layers and recurrent layers of the network, respectively.

The network was trained for 10 epochs using Adam optimizer [46] with a learning rate of $10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$. The minibatch size of 32 was used for training. The model was validated on the validation set every 100 training steps and the loss weights were also updated at the same time. The parameter $W$ (i.e., the window for line fitting and the warm-up period) in Algorithm 1 was set to 20 evaluation steps. We also smoothed the loss curves with $W$-point moving average before line fitting. For the larger databases Physio2018 and SHHS, the training process was early stopped after 200 validation times if no accuracy improvement was recorded on the validation set. Early stopping was not exercised for other databases.

### 4.3 Baseline systems

To assess the efficacy of XSleepNet1 and XSleepNet2, in addition to relevant prior works, we developed three baseline systems for performance comparison. (1) The first two baselines, *FCNN+RNN* and *ARNN+RNN*, are equivalent to the raw and time-frequency network streams shown in Fig. 2, respectively, but they were trained separately. Note that the *ARNN+RNN* is similar to SeqSleepNet presented in [8]. The third baseline, denoted as *Naive Fusion*, combines the multi-view features in a concatenate fashion (i.e., without adaptive gradient blending) and shares a similar network architecture to that of XSleepNet.

The networks were evaluated using five metrics, including accuracy, macro F1-score (MF1) [47], Cohen's kappa [48], average sensitivity, and average specificity. The outputs of a system across the cross-validation folds were pooled altogether to compute its performance metrics. Statistics of the performance metrics, including mean and standard deviation, were also reported in Appendix B.

### 4.4 Experimental results

To give an overview of the performances of XSleepNet1, XSleepNet2, and the baselines, we collate and contrast their overall performances across all the experimental databases across different channel combinations in Fig. 4.

Fig. 4 reveals several compelling patterns. First, between the two single-view baselines, *ARNN+RNN* and *FCNN+RNN*, the former often results in better performance

when the data size is relatively small, for example in SleepEDF-20 and MASS. The opposite is commonly seen with larger databases, such as in Physio2018 and SHHS (EEG). These patterns can be partly explained by the difference in their model footprints: the *ARNN+RNN* with single-channel EEG input has about $1.6 \times 10^5$ parameters in total, 35 times fewer than $5.6 \times 10^6$ parameters in the *FCNN+RNN*. As a result, *FCNN+RNN* is prone to overfitting on the smaller databases (e.g. SleepEDF-20) whereas *ARNN+RNN* is less capable of dealing with the larger databases (e.g. Physio2018). However, it should be stressed that model size is not the only explanation. Indeed, the patterns in SHHS (EEG·EOG and EEG·EOG·EMG) seem to be counterintuitive. As indicated in Section 3.1, the CNN-based and RNN-based models tend to capture different kinds of patterns from their input; therefore, performance discrepancies are expected on different sleep stages [8], [50]. This suggests that how well an individual model performs also depends on the sleep structure of a target cohort and/or the channel combination used.

Second, simple concatenation of the two views in *Naive Fusion* leads to different, potentially diverging results. One can observe clear performance improvements over the two single-view baselines in some cases, such as MASS (EEG) and Physio2018 (EEG). In other cases, such as MASS (EEG·EOG) and Physio2018 (EEG·EOG), *Naive Fusion*'s performance appears to be averaged between the two single-view baselines. There are a few extreme cases where *Naive Fusion* is inferior to both single-view baselines, such as in SleepEDF-20 (EEG) and SleepEDF-78 (EEG·EOG). These diverging patterns indicate that a naive fusion strategy cannot guarantee performance gain, potentially owing to the asynchronous learning behavior of the two views.

Third, via the generalization-/overfitting-aware learning scheme, the XSleepNets can coordinate the learning pace of the views, consolidate the representation power of the views in the joint representation, and convert them into performance gains. Apart from XSleepNet1's modest performance on the smallest database SleepEDF-20 (which will be further discussed), improvements over both the single-view and the naive fusion baselines are consistently seen across all the experimental databases and channel combinations.

In Table 2, we lay out the detailed performance of XSleepNet1, XSleepNet2, the developed baselines, and existing works that used the experimental databases. Particularly, for SleepEDF-20 and SleepEDF-78, we experimented with two common ways these databases were used in literature: (1) only *in-bed* parts of the recordings were used as recommended in [54], [55]; (2) 30 minutes of data before and after in-bed parts were further included in the experiments following the initiation in [20].

There are a few important points. First, the results in Table 2 indicate that the single-view baselines developed in
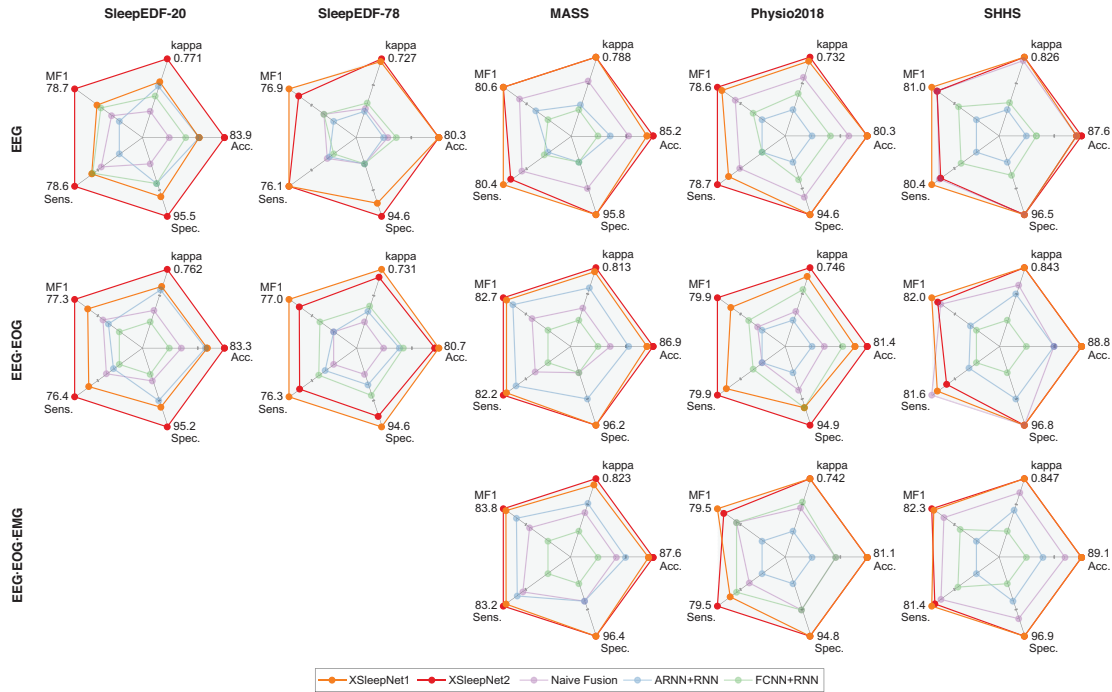
Fig. 4: An overview on the overall performance obtained by XSleepNet1, XSleepNet2, and the developed baselines.

this work are competent models for sleep staging. The rationale is that they all adhere to the state-of-the-art sequence-to-sequence sleep staging framework [8], [50]. Under a similar experimental condition, *ARNN+RNN* outperforms all the existing works across all the scenarios, except U-time [49] on SleepEDF-20. Although the other single-view baseline, *FCNN+RNN*, underperforms on the smallest database, SleepEDF-20, likely owing to its large model footprint, its performance improves when the data size increases. On all the databases but SleepEDF-20, its performance are often better than, and occasionally comparable to, those of the prior works. Second, the results in Table 2 confirm the efficacy of XSleepNet1 and XSleepNet2. They outperform not only the baselines but also prior works across all the databases and channel combinations. On average, XSleepNet1 improves the overall accuracy by $0.9\%$, $0.6\%$, and $1.1\%$ absolute over the *Naive Fusion*, *ARNN+RNN*, and *FCNN+RNN* baselines, respectively. The corresponding accuracy gains are even higher with XSleepNet2, reaching $1.2\%$, $0.9\%$, and $1.4\%$, respectively. Although we do not compute such a performance gain over previous works covered in Table 2 due to incomplete reported results, improvements with large margins can be expected. To demonstrate that XSleepNet also performs well in sleep transfer learning tasks, we carried out the transfer learning tasks as proposed in [50]. More specifically, we pretrained XSleepNet on the MASS database (i.e., the source domain data) (190 subjects used for training and 10 subjects used for validation) and finetuned the pretrained network on the SleepEDF-20 database (i.e., the target domain). Of note, XSleepNet2's approximation approach was used in pretraining whereas XSleepNet1's approximation approach was used in finetuning. This was because we observed that finetuning could converge quickly after a few iterations, XSleepNet2's approximation was inappropriate due to its warmup period. This system is denoted as FT-XSleepNet in Table 2. As

can be seen, FT-XSleepNet outperforms the two transfer-learning approaches proposed in [50], FT-SeqSleepNet+ and FT-DeepSleepNet+, improving the overall accuracy by $0.5\%$ and $1.3\%$ on SleepEDF-20 (EEG) and by $1.6\%$ and $1.3\%$ on SleepEDF-20 (EEG·EOG), respectively.

Inspection on the class-wise performances also reveals that XSleepNet1 and XSleepNet2 lead improvements over the baselines in many databases and channel combinations. More importantly, oftentimes these improvements are spread over the five sleep stages rather than biasing towards some particular ones (see Appendix C).

## 5 DISCUSSION

As mentioned in Section 3.2, XSleepNet2 is devised to overcome the unstable learning behavior of XSleepNet1, especially when the data size is small. Indeed, the stability of the two approaches can be illuminated via their performances shown in Table 2. Both XSleepNet1 and XSleepNet2 work remarkably well when the data size is medium or large (i.e., all the databases but SleepEDF-20) and the latter's performance appears to be better than or equivalent to that of the former in most cases, except for MASS (EEG·EOG·EMG). Their discrepancy becomes clear when inspecting their performance on the smaller database, SleepEDF-20. In this case, although XSleepNet1 still maintains its performance advantages over the *Naive Fusion* baseline, it does not necessarily outperform the best small-footprint baseline *ARNN+RNN*, particularly when its overall accuracy falls short of the baseline's with a gap of $0.8\%$ on SleepEDF-20 (EEG·EOG). In contrast, XSleepNet2's superiority is ubiquitous, not only over all the baselines but also over the XSleepNet1. XSleepNet1's instability is most likely due to the noisy loss weights (cf. Fig. 5) produced by its first-order approach used in generalization/overfitting measure approximation. One possibility to overcome this instability is to utilize the multiple outputs of the network in a self-ensemble fashion (see Appendix D).

TABLE 2: Performance comparison between XSleepNet1, XSleepNet2, the baselines, and previous works on the experimental databases. Of note, the *ARNN+RNN* baseline is equivalent to SeqSleepNet presented in [8]. In addition, the results indicated by † are not directly comparable since they either relied on transfer learning with a pretrained model or used a different subset of the corresponding databases. We include them here for the sake of completeness. We mark in bold where XSleepNets' performances are better or equal to those of all the baselines and compatible prior works.

| Database | System | EEG | | | | | EEG·EOG | | | | | EEG·EOG·EMG | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | κ | MF1 | Sens. | Spec. | Acc. | κ | MF1 | Sens. | Spec. | Acc. | κ | MF1 | Sens. | Spec. |
| SleepEDF-20 (± 30 mins) | **XSleepNet2** | **86.3** | **0.813** | **80.6** | **80.2** | **96.4** | **86.4** | **0.813** | **80.9** | **79.9** | **96.2** | – | – | – | – | – |
| | **XSleepNet1** | **86.0** | **0.810** | **80.0** | **79.6** | **96.3** | 85.2 | 0.798 | **79.8** | 79.0 | 95.9 | – | – | – | – | – |
| | *Naive Fusion* | *85.0* | *0.795* | *78.8* | *78.3* | *96.0* | *83.4* | *0.773* | *77.8* | *77.1* | *95.5* | – | – | – | – | – |
| | *ARNN+RNN (SeqSleepNet [8])* | *85.2* | *0.798* | *78.4* | *78.0* | *96.1* | *86.0* | *0.809* | *79.7* | *79.2* | *96.2* | – | – | – | – | – |
| | *FCNN+RNN* | *81.8* | *0.754* | *75.6* | *75.7* | *95.3* | *83.5* | *0.775* | *77.7* | *77.2* | *95.5* | – | – | – | – | – |
| | DeepSleepNet [20] | – | – | – | – | – | 82.0 | 0.760 | 76.9 | – | – | – | – | – | – | – |
| | U-time [49] | – | – | 79.0 | – | – | – | – | – | – | – | – | – | – | – | – |
| | IITNet [26] | 83.9 | 0.780 | 77.6 | – | – | – | – | – | – | – | – | – | – | – | – |
| SleepEDF-20 | **XSleepNet2** | **83.9** | **0.771** | **78.7** | **78.6** | **95.5** | **83.3** | **0.762** | **77.3** | **76.4** | **95.2** | – | – | – | – | – |
| | **XSleepNet1** | **82.2** | **0.750** | **76.4** | **76.8** | **95.2** | **82.3** | **0.747** | **76.1** | **75.2** | **94.9** | – | – | – | – | – |
| | *Naive Fusion* | *80.2* | *0.723* | *74.9* | *75.8* | *94.7* | *80.8* | *0.726* | *74.7* | *73.7* | *94.5* | – | – | – | – | – |
| | *ARNN+RNN (SeqSleepNet [8])* | *82.2* | *0.746* | *74.1* | *73.9* | *95.0* | *82.2* | *0.744* | *74.2* | *73.1* | *94.8* | – | – | – | – | – |
| | *FCNN+RNN* | *81.3* | *0.737* | *76.0* | *76.7* | *95.0* | *80.1* | *0.716* | *73.2* | *72.6* | *94.4* | – | – | – | – | – |
| | Multitask CNN [19] | 81.9 | 0.740 | 73.8 | – | – | 82.3 | 0.750 | 74.7 | – | – | – | – | – | – | – |
| | DeepSleepNet [20], [50] | 80.8 | 0.731 | 74.2 | – | – | 81.9 | 0.744 | 75.2 | – | – | – | – | – | – | – |
| | 1-max CNN [16] | 79.8 | 0.720 | 72.0 | – | – | – | – | – | – | – | – | – | – | – | – |
| | Attentional RNN [17] | 79.1 | 0.700 | 69.8 | – | – | – | – | – | – | – | – | – | – | – | – |
| | Auto-encoder [21] | 78.9 | – | 73.3 | – | – | – | – | – | – | – | – | – | – | – | – |
| | ResNet [23] | – | 0.650 | – | – | – | – | 0.680 | – | – | – | – | – | – | – | – |
| | VGG-FE [51] | 76.3 | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| | CNN [15] | 74.8 | – | 69.8 | – | – | – | – | – | – | – | – | – | – | – | – |
| | **FT-XSleepNet†** | **85.7** | **0.797** | **80.8** | **80.6** | **96.0** | **85.9** | **0.800** | **80.7** | **81.0** | **96.1** | – | – | – | – | – |
| | FT-SeqSleepNet+† [50] | 85.2 | 0.789 | 79.6 | – | – | 84.3 | 0.776 | 77.7 | – | – | – | – | – | – | – |
| | FT-DeepSleepNet+† [50] | 84.4 | 0.781 | 78.8 | – | – | 84.6 | 0.782 | 79.0 | – | – | – | – | – | – | – |
| | Person. CNN† [25] | 84.0 | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| | VGG-FT† [51] | 80.3 | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| SleepEDF-78 (± 30 mins) | **XSleepNet2** | **84.0** | **0.778** | **77.9** | **77.5** | **95.7** | **84.0** | **0.778** | **78.7** | **77.6** | **95.7** | – | – | – | – | – |
| | **XSleepNet1** | **83.6** | **0.773** | **77.8** | **77.7** | **95.7** | **84.0** | **0.777** | **78.4** | 77.1 | **95.6** | – | – | – | – | – |
| | *Naive Fusion* | *82.3* | *0.755* | *76.2* | *75.7* | *95.3* | *82.5* | *0.757* | *76.9* | *75.8* | *95.3* | – | – | – | – | – |
| | *ARNN+RNN (SeqSleepNet [8])* | *82.6* | *0.760* | *76.4* | *76.3* | *95.4* | *83.8* | *0.776* | *78.2* | *77.4* | *95.6* | – | – | – | – | – |
| | *FCNN+RNN* | *82.8* | *0.761* | *76.6* | *75.9* | *95.4* | *82.7* | *0.759* | *76.9* | *75.5* | *95.3* | – | – | – | – | – |
| | U-Time [49] | – | – | 76.0 | – | – | – | – | – | – | – | – | – | – | – | – |
| | CNN-LSTM [49] | – | – | 73.0 | – | – | – | – | – | – | – | – | – | – | – | – |
| | SleepEEGNet [52] | 80.0 | 0.730 | 73.6 | – | – | – | – | – | – | – | – | – | – | – | – |
| SleepEDF-78 | **XSleepNet2** | **80.3** | **0.727** | **76.4** | **76.1** | **94.6** | **80.6** | **0.728** | **76.7** | **75.8** | **94.5** | – | – | – | – | – |
| | **XSleepNet1** | **80.3** | **0.726** | **76.9** | **76.1** | **94.5** | **80.7** | **0.731** | **77.0** | **76.3** | **94.6** | – | – | – | – | – |
| | *Naive Fusion* | *79.1* | *0.709* | *75.1* | *74.3* | *94.2* | *79.3* | *0.711* | *75.7* | *74.2* | *94.1* | – | – | – | – | – |
| | *ARNN+RNN (SeqSleepNet [8])* | *79.0* | *0.708* | *74.6* | *74.2* | *94.2* | *79.7* | *0.715* | *75.7* | *74.6* | *94.2* | – | – | – | – | – |
| | *FCNN+RNN* | *79.3* | *0.711* | *75.1* | *74.0* | *94.2* | *79.8* | *0.717* | *76.1* | *74.9* | *94.3* | – | – | – | – | – |
| | Personalized SeqSleepNet† [53] | 79.6 | 0.706 | 73.0 | 71.8 | 94.2 | – | – | – | – | – | – | – | – | – | – |
| | DeepSleepNet [20], [53] | 78.5 | 0.702 | 75.3 | 75.0 | 94.1 | – | – | – | – | – | – | – | – | – | – |
| MASS | **XSleepNet2** | **85.2** | **0.788** | **80.6** | **80.2** | **95.8** | **86.9** | **0.813** | **82.7** | **82.2** | **96.2** | **87.6** | **0.823** | **83.8** | **83.2** | **96.4** |
| | **XSleepNet1** | **85.1** | **0.788** | **80.6** | **80.4** | **95.8** | **86.8** | **0.812** | **82.6** | **82.1** | **96.2** | **87.5** | **0.821** | **83.7** | **83.1** | **96.4** |
| | *Naive Fusion* | *84.8* | *0.783* | *80.2* | *79.9* | *95.7* | *86.2* | *0.803* | *81.8* | *81.2* | *96.0* | *86.8* | *0.812* | *82.8* | *82.5* | *96.2* |
| | *ARNN+RNN (SeqSleepNet [8])* | *84.5* | *0.778* | *79.8* | *79.2* | *95.6* | *86.5* | *0.808* | *82.4* | *81.8* | *96.1* | *87.0* | *0.815* | *83.3* | *82.7* | *96.2* |
| | *FCNN+RNN* | *84.3* | *0.777* | *79.5* | *79.3* | *95.6* | *86.0* | *0.800* | *81.3* | *80.8* | *96.0* | *86.4* | *0.806* | *82.1* | *81.6* | *96.1* |
| | DeepSleepNet [8], [20] | – | – | – | – | – | – | – | – | – | – | 86.4 | 0.805 | 82.2 | 81.8 | 96.1 |
| | Multitask CNN [8], [19] | – | – | – | – | – | – | – | – | – | – | 83.6 | 0.766 | 77.9 | 77.4 | 95.3 |
| | Attentional RNN [8], [17] | – | – | – | – | – | – | – | – | – | – | 83.6 | 0.766 | 78.4 | 78.0 | 95.3 |
| | 1-max CNN [8], [16] | – | – | – | – | – | – | – | – | – | – | 82.7 | 0.754 | 77.6 | 77.8 | 95.1 |
| | CNN [8], [18] | – | – | – | – | – | – | – | – | – | – | 79.9 | 0.726 | 76.7 | 80.0 | 95.0 |
| | CNN [8], [15] | – | – | – | – | – | – | – | – | – | – | 77.9 | 0.680 | 70.4 | 69.4 | 93.5 |
| | ResNet [23] | – | 0.670 | – | – | – | – | 0.720 | – | – | – | – | 0.740 | – | – | – |
| | IITNet† [26] | 86.3 | 0.790 | 80.5 | – | – | – | – | – | – | – | – | – | – | – | – |
| | DeepSleepNet† [20] | – | – | – | – | – | 86.2 | 0.800 | 81.7 | – | – | – | – | – | – | – |
| | DNN+RNN† [22] | – | – | – | – | – | 85.9 | – | 80.5 | – | – | – | – | – | – | – |
| | DNN† [22] | – | – | – | – | – | 81.6 | – | 77.2 | – | – | – | – | – | – | – |
| Physio2018 | **XSleepNet2** | **80.3** | **0.732** | **78.6** | **78.7** | **94.6** | **81.4** | **0.746** | **79.9** | **79.9** | **94.9** | **81.1** | **0.742** | **79.4** | **79.5** | **94.8** |
| | **XSleepNet1** | **80.3** | **0.731** | **78.5** | **78.4** | **94.6** | **81.2** | **0.744** | **79.6** | **79.7** | **94.8** | **81.1** | **0.742** | **79.5** | **79.3** | **94.8** |
| | *Naive Fusion* | *80.0* | *0.727* | *78.2* | *78.1* | *94.5* | *80.7* | *0.736* | *79.0* | *78.9* | *94.7* | *80.7* | *0.737* | *79.2* | *79.0* | *94.7* |
| | *ARNN+RNN (SeqSleepNet [8])* | *79.4* | *0.719* | *77.6* | *77.5* | *94.3* | *80.5* | *0.734* | *78.9* | *78.9* | *94.6* | *80.4* | *0.733* | *78.8* | *78.8* | *94.6* |
| | *FCNN+RNN* | *79.7* | *0.723* | *77.8* | *77.5* | *94.4* | *81.0* | *0.741* | *79.2* | *79.1* | *94.8* | *80.7* | *0.738* | *79.2* | *79.2* | *94.7* |
| | U-Time [49] | – | – | 77.0 | – | – | – | – | – | – | – | – | – | 77.0 | – | – |
| | CNN-LSTM [49] | – | – | 77.0 | – | – | – | – | – | – | – | – | – | – | – | – |
| SHHS | **XSleepNet2** | **87.6** | **0.826** | 80.7 | 79.7 | **96.5** | **88.8** | **0.843** | **81.8** | 80.8 | **96.8** | **89.1** | **0.847** | **82.3** | 81.2 | **96.9** |
| | **XSleepNet1** | **87.5** | **0.826** | **81.0** | **80.4** | **96.5** | **88.8** | **0.843** | **82.0** | 81.3 | **96.8** | **89.1** | **0.847** | **82.2** | **81.4** | **96.9** |
| | *Naive Fusion* | *87.5* | *0.825* | *80.7* | *79.8* | *96.5* | *88.4* | *0.839* | *81.7* | *81.6* | *96.8* | *88.8* | *0.843* | *81.7* | *80.8* | *96.8* |
| | *ARNN+RNN (SeqSleepNet [8])* | *86.5* | *0.811* | *78.5* | *76.9* | *96.1* | *88.4* | *0.837* | *80.7* | *79.6* | *96.7* | *88.4* | *0.838* | *80.1* | *78.5* | *96.7* |
| | *FCNN+RNN* | *86.7* | *0.813* | *79.5* | *78.1* | *96.2* | *88.0* | *0.831* | *80.5* | *79.2* | *96.6* | *88.1* | *0.832* | *80.9* | *79.7* | *96.6* |
| | CNN [24] | 86.8 | 0.810 | 78.5 | – | 95.0 | – | – | – | – | – | – | – | – | – | – |
| | IITNet [26] | 86.7 | 0.810 | 79.8 | – | – | – | – | – | – | – | – | – | – | – | – |

To see how the loss weights were adapted for gradient blending during the training course of the XSleepNets, we take one cross-validation fold of the MASS database as an example (illustrated in Fig. 5). In case of the single-view training with *ARNN+RNN* and *FCNN+RNN*, even though the former's validation loss did not show signs of overfitting, the latter's with a large model footprint converged and started overfitting after $10^5$ training steps (cf. Fig. 5 (a)). As a consequence, when the two views were trained jointly with a simple fusion strategy in *Naive Fusion*, the overfitting network stream kept learning at the same rate, impairing the joint-view representation and causing overfitting (even though less serious) in the joint classification branch (it can be roughly thought as being the average of the two views). In contrast, during the training of XSleepNet1 (cf. Fig. 5 (b) and (d)), the view in red that was converging faster was initially associated with an increasingly large weight while a small weight was assigned to the one in blue (the slower one). Note that the view in blue was still able to learn due to the gradient flow coming from the joint classification branch (the green curve). The turning point was when the view in red converged and started overfitting shortly afterwards; its weights descended, impeding the learning and preventing it from overfitting the data at the regular pace. The decreasing weight of the view in red was gradually transferred to and accelerate the learning of the view in blue. As a result, the joint learning process yields the joint representation which is more robust to overfitting and more generalizable than that learned by the *Naive Fusion* baseline. On the other hand, XSleepNet2's losses and weights (cf. Fig. 5 (c) and (e)) reveal different learning behavior from that of XSleepNet1. Apart from being much smoother, the loss weights appear to be adapted to synchronize the learning pace of the two views and then maintain this synchronization throughout rather than behaving in an turn-taking fashion as in XSleepNet1.

Out of five experimental databases, only on SHHS, the largest database, did we see similar behavior between the *Naive Fusion* and the XSleepNets. In other words, the *Naive Fusion* model resulted in consistently better results than the single-view baselines (cf. Fig. 4 and Table 2). The reason is when the data becomes large enough, it imposes a strong regularization on the network. As a consequence, the networks, either in single-view training or in joint training, did not experience overfitting as with smaller databases, evidenced by the validation losses in Fig. 5. We anticipate that the model size can be safely increased in this case.

It is worth mentioning that even though XSleepNet has two network streams, its model footprint ($5.8 \times 10^6$ parameters) is still 4 times smaller than it of the popular DeepSleepNet ($22.9 \times 10^6$ parameters). In addition, utilizing multiple EEG channels, when available, of a sleep database would further improve XSleepNet's [18]. An interesting question is whether to combine the EEG channels as in [18] or consider them as different views and using our proposed muliti-view approach for channel combination. Although we experimented the XSleepNets on automatic sleep staging with PSG signals, the method is generic enough to serve sleep analysis with other modalities, especially when multimodal data are available [56], [57]. It is also applicable to other applications where the target signals are inherently multi-view. One example is audio/speech in which raw audio signals [58]
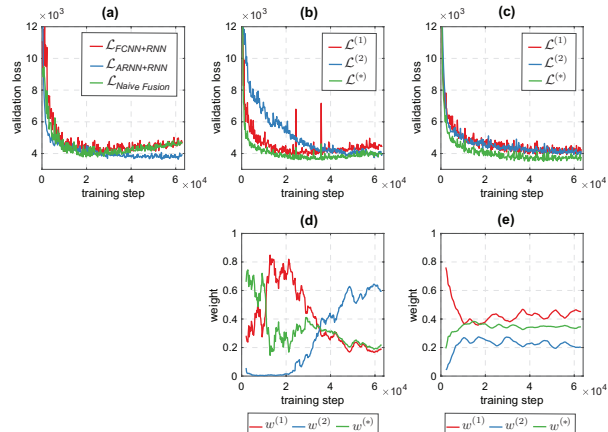


Fig. 5: Progression of the validation losses and the adaptive loss weights during the training course of one MASS cross-validation fold. (a) The validation losses of the *FCNN+RNN*, *ARNN+RNN*, and *Naive Fusion* baselines; (b) The validation losses of the three classification branches of XSleepNet1 and (d) their respective adaptive loss weights; (c) The validation losses of the three classification branches of XSleepNet2 and (e) their respective adaptive loss weights. It should be emphasized that the adaptive loss weights in (c) were denoised with a 10-point moving average filter before plotting; the original ones are expectedly much noisier.
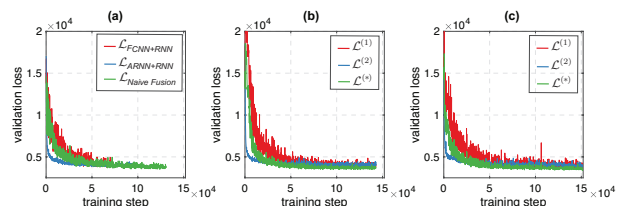


Fig. 6: The validation losses during the training course of the SHHS database. (a) The validation losses of the *FCNN+RNN*, *ARNN+RNN*, and *Naive Fusion* baselines; (b) The validation losses of the three classification branches of XSleepNet1; (c) The validation losses of the three classification branches of XSleepNet2.

can be combined with its derived representations, such as mel-scale spectrogram [59] and gammatone spectrogram [60], for recognition tasks. Another example is computer vision in which different image channels, such as luminance, chrominance, depth, and optical flow, are essentially multi-view data [28], [61].

## 6 CONCLUSIONS

In this paper, we presented XSleepNet, a sequence-to-sequence network architecture for automatic sleep staging, that is capable of learning from both raw signal and time-frequency input at the same time. The network architecture accommodates two network streams, one for each input view. XSleepNet is principally designed to be robust to training data size, complementary between the constituent network streams, and aware of generalization and overfitting behavior of its network streams. The network can be trained in such a way that learning on the generalizing network stream is encouraged while that on the overfitting one is discouraged. Two approaches were introduced to

approximate generalization/overfitting measure on classification branches of the network and produce the respective weights for gradient blending, resulting in two models, XSleepNet1 and XSleepNet2. By regulating the weights, and hence the learning pace of the network streams, XSleepNet1 and XSleepNet2 yielded joint features which represent the underlying data distribution better than those learned by the single-view baselines as well as the multi-view baseline following a naive fusion approach. Empirical evaluation showed that they not only delivered more favorable results than the baselines but also outperformed existing work on five databases of different sizes. Between the two models, XSleepNet2, relying on a second-order approximation algorithm, resulted in more stable performance than XSleepNet1, using a first-order approximation algorithm.

## ACKNOWLEDGMENTS

## REFERENCES

[1] P. Maquet, "The role of sleep in learning and memory," *Science*, vol. 294, no. 5544, pp. 1048–1052, 2001.

[2] A. C. Krieger, Ed., *Social and Economic Dimensions of Sleep Disorders, An Issue of Sleep Medicine Clinics*, Elsevier, 2017.

[3] V. K. Chattu *et al.*, "The global problem of insufficient sleep and its serious public health implications," in *Healthcare*, 2019, vol. 7.

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

[5] K. B. Mikkelsen *et al.*, "Machine-learning-derived sleep–wake staging from around-the-ear electroencephalogram outperforms manual scoring and actigraphy," *J. Sleep Res.*, vol. 28, no. 2, pp. e12786, 2019.

[6] K. B. Mikkelsen *et al.*, "Accurate whole-night sleep monitoring with dry-contact ear-EEG," *Sci. Rep.*, vol. 9, no. 1, pp. 1–12, 2019.

[7] A. Malhotra *et al.*, "Performance of an automated polysomnography scoring system versus computer-assisted manual scoring," *SLEEP*, vol. 36, no. 4, pp. 573–582, 2013.

[8] H. Phan *et al.*, "SeqSleepNet: end-to-end hierarchical recurrent neural network for sequence-to-sequence automatic sleep staging," *IEEE Trans. Neural Syst. Rehabilitation Eng.*, vol. 27, no. 3, pp. 400–410, 2019.

[9] J. B. Stephansen et al., "Neural network analysis of sleep stages enables efficient diagnosis of narcolepsy," *Nature Communications*, vol. 9, no. 1, pp. 5229, 2018.

[10] C. O'Reilly *et al.*, "Montreal archive of sleep studies: An open-access resource for instrument benchmarking & exploratory research," *J. Sleep Res.*, pp. 628–635, 2014.

[11] S. Biswal *et al.*, "Expert-level sleep scoring with deep neural networks," *J Am Med Inform Assoc.*, vol. 25, no. 12, pp. 1643–1650, 2018.

[12] S. Biswal *et al.*, "SLEEPNET: Automated sleep staging system via deep learning," *arXiv preprint arXiv:1707.08262*, 2017.

[13] H. Sun *et al.*, "Large-scale automated sleep staging," *SLEEP*, vol. 40, no. 10, pp. zsx139, 2017.

[14] M. Längkvist, L. Karlsson, and A. Loutfi, "Sleep stage classification using unsupervised feature learning," *Advances in Artificial Neural Systems*, vol. 2012, pp. 1–9, 2012.

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

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

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

[18] S. Chambon *et al.*, "A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series," *IEEE Trans. Neural Syst. Rehabilitation Eng.*, vol. 26, no. 4, pp. 758–769, 2018.

[19] H. Phan *et al.*, "Joint classification and prediction CNN framework for automatic sleep stage classification," *IEEE. Trans. Biomed. Eng.*, vol. 66, no. 5, pp. 1285–1296, 2019.

[20] 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. Rehabilitation Eng.*, vol. 25, no. 11, pp. 1998–2008, 2017.

[21] O. Tsinalis *et al.*, "Automatic sleep stage scoring using time-frequency analysis and stacked sparse autoencoders," *Ann Biomed Eng*, vol. 44, no. 5, pp. 1587–1597, 2016.

[22] H. Dong *et al.*, "Mixed neural network approach for temporal sleep stage classification," *IEEE Trans. Neural Syst. Rehabilitation Eng.*, vol. 26, no. 2, pp. 324–333, 2018.

[23] F. Andreotti *et al.*, "Multichannel sleep stage classification and transfer learning using convolutional neural networks," in *Proc. EMBC*, 2018, pp. 171–174.

[24] A. Sors *et al.*, "A convolutional neural network for sleep stage scoring from raw single-channel eeg," *Biomed Signal Process Control*, vol. 42, pp. 107–114, 2018.

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

[26] H. Seo *et al.*, "Intra- and inter-epoch temporal context network (iitnet) using sub-epoch features for automatic sleep scoring on raw single-channel EEG," *Biomed Signal Process Control*, vol. 61, pp. 102037, 2020.

[27] J. Zhao *et al.*, "Multi-view learning overview: Recent progress and new challenges," *Information Fusion*, vol. 38, pp. 43–54, 2017.

[28] X. Chen *et al.*, "Multi-view 3d object detection network for autonomous driving," in *Proc. CVPR*, 2017, pp. 6526–6534.

[29] A. Tsymbal, M. Pechenizkiy, and P. Cunningham, "Diversity in search strategies for ensemble feature selection," *Information Fusion*, vol. 6, no. 1, pp. 83–98, 2005.

[30] H. Phan *et al.*, "Fusion of end-to-end deep learning models for sequence-to-sequence sleep staging," in *Proc. EMBC*, 2019, pp. 1829–1833.

[31] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.

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

[33] T. Cooijmans *et al.*, "Recurrent batch normalization," in *ICLR (Poster)*, 2017.

[34] J. Long *et al.*, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, 2015, pp. 3431–3440.

[35] K. He *et al.*, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. ICCV*, 2015, pp. 1026–1034.

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

[37] J. A. Hobson, "A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects," *Electroencephalography and Clinical Neurophysiology*, vol. 26, no. 6, pp. 644, 1969.

[38] W. Wang, D. Tran, and M. Feiszli, "What makes training multi-modal networks hard?," in *Proc. CVPR*, 2020, pp. 12692–12702.

[39] B. Kemp *et al.*, "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, 2000.

[40] A. L. Goldberger *et al.*, "Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, pp. e215–e220, 2000.

[41] M. M. Ghassemi *et al.*, "You snooze, you win: the physionet/computing in cardiology challenge 2018," in *Proc. 2018 Computing in Cardiology Conference (CinC)*, 2018, vol. 45.

[42] G. Q. Zhang *et al.*, "The national sleep research resource: towards a sleep data commons," *J Am Med Inform Assoc.*, vol. 25, no. 10, pp. 1351–1358, 2018.

[43] S. F. Quan *et al.*, "The sleep heart health study: design, rationale, and methods," *Sleep*, vol. 20, no. 12, pp. 1077–1085, 1997.

[44] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," in *Proc. OSDI*, 2016, pp. 265–283.

[45] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *J Mach Learn Res*, vol. 15, pp. 1929–1958, 2014.

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

[47] Y. Yang, , and X. Liu, "A re-examination of text categorization methods," in *Proc. SIGIR*, 1999, vol. 99, pp. 42–49.

[48] M. L. McHugh, "Interrater reliability: the kappa statistic," *Biochemia Medica*, vol. 22, no. 3, pp. 276–282, 2012.

[49] M. Perslev *et al.*, "U-time: A fully convolutional network for time series segmentation applied to sleep staging," in *Proc. NeurIPS*, 2019, pp. 4417–4428.

[50] H. Phan *et al.*, "Towards more accurate automatic sleep staging via deep transfer learning," *IEEE Trans. Biomed. Eng.*, 2020.

[51] A. Vilamala, K. H. Madsen, and L. K. Hansen, "Deep convolutional neural networks for interpretable analysis of EEG sleep stage scoring," in *Proc. MLSP*, 2017, pp. 1–6.

[52] S. Mousavi, F. Afghah, and R. Acharya, "SleepEEGNet: Automated sleep stage scoring with sequence to sequence deep learning approach," *PLoS One*, vol. 14, no. 5, pp. e0216456, 2019.

[53] H. Phan *et al.*, "Personalized automatic sleep staging with single-night data: a pilot study with KL-divergence regularization," *Physiological Measurement*, vol. 41, no. 6, pp. 064004, 2020.

[54] S. A. Imtiaz and E. Rodriguez-Villegas, "Recommendations for performance assessment of automatic sleep staging algorithms," in *Proc. EMBC*, 2014, pp. 5044–5047.

[55] S. A. Imtiaz and E. Rodriguez-Villegas, "An open-source toolbox for standardized use of PhysioNet Sleep EDF Expanded Database," in *Proc. EMBC*, 2015, pp. 6014–6017.

[56] B. Zhai *et al.*, "Making sense of sleep: Multimodal sleep stage classification in a large, diverse population using movement and cardiac sensing," in *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2020, vol. 4.

[57] A. N. Olesen *et al.*, "Towards a flexible deep learning method for automatic detection of clinically relevant multi-modal events in the polysomnogram," in *Proc. EMBC*, 2019, pp. 556–561.

[58] D. Palaz *et al.*, "Convolutional neural networks-based continuous speech recognition using raw speech signal," in *Proc. ICASSP*, 2015, pp. 4295–4299.

[59] D. Amodei *et al.*, "Deep speech 2 : End-to-end speech recognition in english and mandarin," in *Proc. ICML*, 2016, pp. 173–182.

[60] R. Schlüter *et al.*, "Gammatone features and feature combination for large vocabulary speech recognition," in *Proc. ICASSP*, 2007, pp. 649–652.

[61] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *Proc. ECCV*, 2020, pp. 776–794.
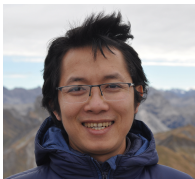
**Huy Phan** received the M.Eng. degree from Nanyang Technological University, Singapore, in 2012, and the Dr.-Ing. degree in computer science from University of Lübeck, Germany, in 2017. From 2017 to 2018, he was a Postdoctoral Research Assistant with University of Oxford, UK. From 2019 to 2020, he was a Lecturer at University of Kent, UK. In April 2020, he joined Queen Mary University of London, UK, where he is a Lecturer in Artificial Intelligence. His research interests include machine learning and signal processing with a special focus on audio and biosignal analysis. In 2018, he received the Bernd Fischer Award for the best PhD thesis from University of Lübeck. In 2021, he was awarded Benelux's IEEE-EMBS Best Paper Award 2019-20.



**Oliver Y. Chén** reads biomedical engineering in the Institute of Biomedical Engineering at University of Oxford and is an Honorary Research Fellow in the Division of Biosciences at University College London. He received his Master's in Biostatistics from Johns Hopkins University, Master's in Mathematical Statistics from Washington State University, and Ph.D. in Engineering Science at University of Oxford. He was a Research Fellow in Neuroscience and Psychology at Yale University; he gave the Si-Shui Lecture and received the Louise I. and Thomas D. Dublin Award for the Advancement of Epidemiology and Biostatistics.



**Minh C. Tran** received the B.Eng. (Hons) degree in Bioengineering from the University of Sheffield, United Kingdom in 2017. He is currently a graduate reading for D.Phil. in Engineering Science at the University of Oxford. He is a member of the Nuffield Division of Anaesthetics, University of Oxford. His research interests include biosignal modelling and processing, respiratory and intensive care medical devices, and entrepreneurship.



**Philipp Koch** received the B.Sc. and M.Sc. degrees in medical engineering science in 2013 and 2015, respectively, from the University of Lübeck, Lübeck, Germany, where he is currently working toward the Ph.D. degree. He is a Research Associate at the Institute for Signal Processing, University of Lübeck. His research interests include machine learning, biosignal analysis, audio/acoustic signal processing, and sparse MRI reconstruction.



**Alfred Mertins** received the Dipl.-Ing. degree from the University of Paderborn, Paderborn, Germany, in 1984, the Dr.-Ing. degree in electrical engineering and the Dr.-Ing. Habil. degree in telecommunications from Hamburg University of Technology, Hamburg, Germany, in 1991 and 1994, respectively. From 1986 to 1991, he was a Research Assistant with Hamburg University of Technology, and from 1991 to 1995, he was a Senior Scientist with the Microelectronics Applications Center Hamburg, Germany. From 1996 to 1997, he was with the University of Kiel, Germany, and from 1997 to 1998, with the University of Western Australia, WA, Australia. In 1998, he joined the University of Wollongong, Wollongong, NSW, Australia, where he was an Associate Professor of electrical engineering. From 2003 to 2006, he was a Professor with the Faculty of Mathematics and Science, University of Oldenburg, Oldenburg, Germany. In November 2006, he joined the University of Lübeck, Lübeck, Germany, where he is a Professor and the Director of the Institute for Signal Processing. His research interests include speech, audio, and image processing; wavelets and filter banks; pattern recognition; and digital communications.



**Maarten De Vos** has a joint appointment as Associate Professor in the Departments of Engineering and Medicine at KU Leuven after being Associate Professor at the University of Oxford, United Kingdom, and Junior Professor at the University of Oldenburg, Germany. He obtained an MSc (2005) and PhD (2009) in Electrical Engineering from KU Leuven, Belgium. His academic work focuses on AI for health, innovative biomedical monitoring and signal analysis for daily life applications, in particular the derivation of personalised biosignatures of patient health from data acquired via wearable sensors and the incorporation of smart analytics into unobtrusive systems. His pioneering research in the field of mobile real-life brain-monitoring has won several innovation prices, among which the prestigious Mobile Brain Body monitoring prize in 2017. In 2019, he was awarded the Martin Black Prize for the best paper in Physiological Measurement.