A Structured Learning Approach with Neural Conditional Random Fields for Sleep Staging

Karan Aggarwal, Swaraj Khadanga
University of Minnesota
Minneapolis, MN
{aggar081, khada004}@umn.edu

Shafiq R. Joty
Nanyang Technological University
Singapore
srjoty@ntu.edu.sg

Louis Kazaglis
Fairview Health
Minneapolis, MN
lkazagl1@fairview.org

Jaideep Srivastava
University of Minnesota
Minneapolis, MN
srivasta@umn.edu

***Abstract*—Sleep plays a vital role in human health, both mental and physical. Sleep disorders like sleep apnea are increasing in prevalence, with the rapid increase in factors like obesity. Sleep apnea is most commonly treated with Continuous Positive Air Pressure (CPAP) therapy. Presently, however, there is no mechanism to monitor a patient's progress with CPAP. Accurate detection of sleep stages from CPAP flow signal is crucial for such a mechanism. We propose, for the first time, an automated sleep staging model based only on the flow signal.**

Deep neural networks have recently shown high accuracy on sleep staging by eliminating handcrafted features. However, these methods focus exclusively on extracting informative features from the input signal, without paying much attention to the dynamics of sleep stages in the output sequence. We propose an end-to-end framework that uses a combination of deep convolution and recurrent neural networks to extract high-level features from raw flow signal with a structured output layer based on a conditional random field to model the temporal transition structure of the sleep stages. We improve upon the previous methods by 10% using our model, that can be augmented to the previous sleep staging deep learning methods. We also show that our method can be used to accurately track sleep metrics like sleep efficiency calculated from sleep stages that can be deployed for monitoring the response of CPAP therapy on sleep apnea patients. Apart from the technical contributions, we expect this study to motivate new research questions in sleep science.

***Index Terms*—Sleep Staging, Conditional Random Fields, Structured Prediction**

I. Introduction

Sleep plays a fundamental role in the physical and emotional recovery of the human body. Sleep deprivation or poor quality of sleep adversely affect the quality of life. Outside of the wake state, sleep can be divided into four stages: *Rapid Eye Movement* (REM), and *Non-REM* (NREM) stages 1, 2, and 3 [1]. Due to transitory nature of NREM stage 1, stages 1 and 2 are often grouped and classified as light sleep, as compared to deep sleep for NREM stage 3. Each stage has its role in the recovery process, *e.g.,* REM sleep helps in memory consolidation and emotion regulation while deep sleep helps with physical recovery processes. Understanding of a subject's sleep states and their dynamics is necessary for identifying and monitoring various sleep-related disorders such as sleep apnea and insomnia.

The economic cost of sleep-related disorders is enormous [2]. One of the leading cost burdens is due to *Obstructive Sleep Apnea* (OSA). OSA is a disorder in which airway collapses during inhalation resulting in a reduced oxygen supply to the brain forcing the patient to wake, causing interrupted sleep. OSA poses a severe risk, *e.g.,* OSA is associated with higher rates of heart attacks. Despite the severity of the condition, it is a mostly undiagnosed disease with an estimated 5%-20% prevalence rates among the population [3] with an estimated cost burden of $150 billion per year in the US alone [4].

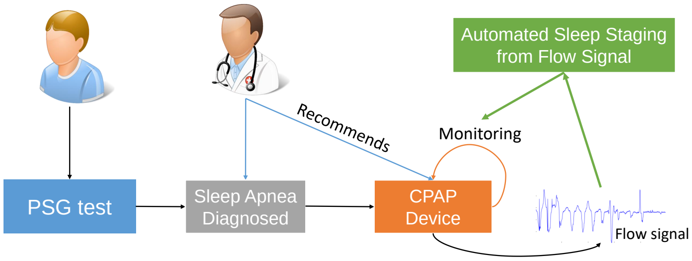


Fig. 1: An application use case of our model. A patient undergoes Polysomnography (PSG) to ascertain the sleep disorders and is diagnosed with Sleep Apnea. Healthcare provider recommends the CPAP therapy that involves a CPAP device. Flow signal can be obtained from the device daily for monitoring purposes. By adding the automated sleep staging step, we can help the healthcare provider with a means for continuous monitoring of the patient.

The most prevalent and effective treatment for OSA is *Continuous Positive Airway Pressure* (CPAP) therapy. In a CPAP therapy, a user wears a mask, connected to a flow generating device, which delivers an adaptive pressure to prevent the collapse of the airway and track signals like daily airflow pressure (*flow signal*) data.This data contains valuable information transmitted to healthcare professionals for monitoring the subject's respiratory patterns. However, it is not being utilized actively to monitor the efficacy of patient therapy or sleep quality. The key to measure the effectiveness of CPAP therapy is to assess the sleep quality by determining the sleep stages. However, to the best of our knowledge, this is the first attempt at determining sleep stages from CPAP-available signals. Determination of sleep stages has been typically performed on data obtained from Polysomnograms (PSG), which involves an overnight measurement of a variety of biological signals during sleep. The gold standard for securing sleep stages is for trained sleep experts to manually annotate PSG data, a tedium-filled expensive task at best.

Prior studies on sleep staging have focused on automating the annotations by using reduced number of sensors from PSG including Electroencephalography (EEG) [5], [6] or using other more comfortable devices like actigraphy [7], cardio-respiratory sensors [8], or no-contact sensors [9]. However, all of these approaches do not have a direct use case - they require additional devices to provide data for sleep staging. In this work, *we make the first attempt to use the CPAP-available flow signal to identify sleep stages automatically*. CPAP users can know about their sleep health by learning about their sleep states, while the health-care providers can track longitudinal sleep health and overall success of CPAP therapy. Figure 1 shows a schematic of the application of our work. A benefit of this study would be to interest the sleep research community in investigating the effect of CPAP therapy on sleep architecture trajectory of OSA patients.

On the technical front, most previous approaches have used hand-crafted features [5], [10] for the task. Recently, deep neural networks [11], [12], [13] have been used for end-to-end learning without manual feature engineering mainly based on convolutional neural networks (CNN). Hybrid recurrent-convolutional neural networks (R-CNN) [14], [15] methods that use CNN as base network fed to the recurrent networks have shown human-expert level accuracy on PSG. Adversarial training with R-CNN proposed by Zhao et al. [9] has shown state-of-the-art results on RF-signals. These methods focus *exclusively on learning informative abstract features* from the input signal making predictions at each time step independent of the previous sleep state. However, sleep states have a strong transition structure [16]. By not taking into account the dynamics of the sleep states, the deep learning methods have missed out on an essential source of information.

In this work, we propose a new neural network architecture based on *chain-structured conditional random field (CRF)* that explicitly models the temporal dynamics in the sleep states, over the *deep convolutional neural network* to learn high-level abstract features from CPAP flow signals and a *recurrent neural network* to encode temporal context in these features. The entire **Neural CRF** (CNN-RNN-CRF) network is trained for sleep staging in an end-to-end fashion.

Our Neural CRF method shows a substantial improvement over the state-of-art when applied to the CPAP flow signal for sleep staging. Further, we improve the performance using a class distribution cost-sensitive prior to deal with the imbalanced distribution of sleep stages and using a domain dependent regularization over the CRF parameters. In summary, we make the following contributions:

(a) While the prior deep learning works have entirely focused on extracting best features from the input signals, we demonstrate that jointly modeling the dynamics of the output sleep stages can substantially increase the performance. We improve over the state-of-the-art conditional adversarial model [9] by over 10% in terms of Cohen's Kappa score. Our approach can be added to the existing deep learning models that are competing in the input space.

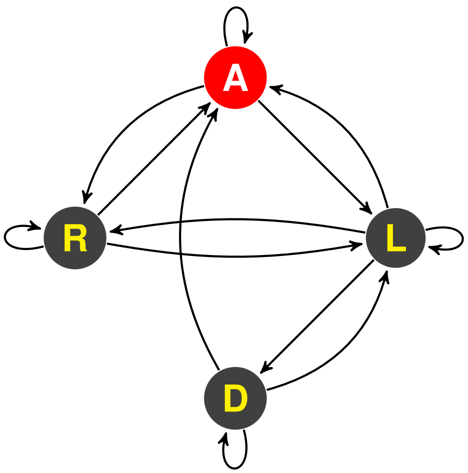


Fig. 2: Transition diagram for OSA patients [16] with non-REM states 1 and 2 combined as Light (L) sleep state. Four sleep states shown are: (A)wake, (R)EM, (L)ight and (D)eep.

(b) We propose to use a CNN architecture along with a recurrent layer to extract high-level features from CPAP flow signals. Our architecture is inspired by the popular ResNet [17] used in computer vision.

(c) We present the first study on automatic sleep staging using the respiratory flow signal. Our model has a direct existing application use case - providing healthcare professionals a way to track the patients undergoing sleep apnea treatment through CPAP devices. By directly linking CPAP flow data to sleep stages, this work has the potential to illustrate an improvement in sleep and create an interest in investigating the cognitive and neuronal benefits of adhering to CPAP therapy.

We organize the rest of the paper as follows. Section II places our work in the context of the existing literature. Section III describes our proposed solution in detail. Section IV lays out our experimental settings. We present our results and analysis in Section V. Finally, we conclude in Section VI.

II. Related Work

In this section, we describe the different approaches that have been taken towards automating the sleep staging process. PSG is the gold standard for assessment of sleep quality and diagnosis of specific sleep disorders. It requires the subject to spend a night in a sleep lab with a variety of sensors attached to collect data about the biological processes during sleep. In clinical practice, several levels of health-care sleep professionals visually annotate the data in 30-second epochs to ascertain the sleep stages. Sleep staging is a labor-intensive process with limitations due to inter-expert variability [18]. There have been some recent efforts to develop automated sleep staging methods. However, the PSG process has a very high overhead in terms of cost and convenience, and so most of the effort has focused on reducing the number of sensors to asses sleep stages as an alternative to PSG. We divide the related literature into two parts in the context of our work: (a) signal sources and their application context, and (b) machine learning based sleep staging models.

***(a)* Signal sources and their application context: EEG-based** sensors measure brain activity and have been shown to indicate sleep states the best, as sleep staging is based on EEG rules during PSGs. EEG-based sleep staging [19], [5], [6] has demonstrated the highest accuracy. However, long-term EEG sensor recording is not practical being both costly and inconvenient. Hence, using more convenient sensors has been proposed. **Cardio-respiratory sensors** based methods [10], [20] utilizing the electrocardiogram (EKG) or cardiac impulse signalshave shown moderate accuracy. **Actigraphy-based** methods [21], [7] to measure movement are good discriminators between wake and sleep but a poor predictor of sleep stages [22]. **No-contact** sensors like smartphone and radio-signal are the most convenient for the user. Smartphone-based approaches [23], [24] have shown to perform poorly like actigraphy.RF-based approaches have historically demonstrated low accuracy, though recent work by Zhao et al. [9] demonstrated significant improvements.

In our work, we use a *new source signal,* namely ***nasal airflow (flow)***. This flow signal is a measure of respiratory effort, similar to chest-band based sensors that measure the respiratory patterns of subjects during sleep. Unlike other methods, however, *the flow signal has an existing use-case*; persons with OSA who are regularly using CPAP therapy. Our method can provide a mechanism for continuously monitoring these patients' sleep health and response to the therapy with significantly improved accuracy for sleep staging and very high accuracy for sleep efficiency. On the clinical research side, we expect that our study motivates researchers for investigating the brain and cognitive effects of CPAP therapy.

***(b)* Machine learning based sleep staging models:** Most of the prior studies using machine learning have used handcrafted features [5], [10], [20]. Features based on frequency domain like power spectral density and time domain like variance, skew, or kurtosis have been used mostly as input to classifiers. Recently, **deep learning** methods using deep convolution neural networks [6], [11], [13], [12] have been proposed. Hybrid R-CNN models with CNN as the base network, followed by a recurrent neural network (RNN) have shown state-of-the-art results [14], [15] comparable to expert level annotations on PSG data. Zhao et al. [9] have proposed an adversarial R-CNN architecture that achieves state-of-the-art results using the radio frequency (RF) signals.

These deep learning methods have entirely focused on extracting the best possible features from the input signal ignoring or paying limited attention to the context of each segment and dynamics of the sleep states. However, sleep stage transitions have a strong dependency structure [16] with many transitions having extremely low probability. For example, the transition between REM and deep sleep requires an intermediate step; having contiguous REM and deep sleep epochs would require the unlikely occurrence of several transitions taking place inside the same epoch. Furthermore, some detectable events like rapid eye movements, arousals or K-spindles [25] dictate epoch-to-epoch stability or stage transitions. In such scenarios, it is essential to take into account both the input signal and the dynamics of sleep states. R-CNN models assume that recurrent connections can capture the sleep state transition structure while attempts with convolution network [12] assume that taking the immediate neighboring segments into account should suffice.

In this work, we use a **conditional random field** model that does joint modeling of the sleep stages for the entire duration of sleep, trained end-to-end with a deep R-CNN network. We show that using this approach we can substantially increase the model's performance over adversarial [9] or baseline R-CNN. Our approach can be augmented to any of the existing deep learning methods.

III. Model

In this section, we present our problem and the proposed solution that combines a convolutional neural network (CNN), a recurrent neural network (RNN), and a conditional random field (CRF) in a single architecture that is trained end-to-end. In the following, we describe the components.

A. Problem Statement

Let the input flow signal time-series be $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where x_i's are signal values sampled at 32 Hertz, *i.e.,* 32 signal values per second. Annotation of sleep stages is done for each 30-second epoch corresponding to $30 \times 32 = 960$ signal values in $\mathbf{x}$. An example flow signal and corresponding sleep stages for a night's sleep is shown in Figure 4. The computational task is to annotate the signal time-series for each 30-second epoch with a label y_i as one of the four sleep stages: **A**wake, **R**EM, **L**ight Sleep, and **D**eep Sleep. In other words, we need to label the sequence $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ with $\mathbf{y} = (y_1, y_2, \ldots, y_m)$, where $m = n/960$. In our experiments, $m = 900$ and $n = 900 \times 960 = 864,000$. We denote the set of sleep states as $K = \{A, R, L, D\}$, in the rest of the paper.

B. Convolutional Neural Network

Convolution neural networks have been used extensively for a variety of tasks in computer vision, natural language processing, and time-series analysis. We use convolution layers to extract high-level abstract features that are then fed to the recurrent layer. Our convolutional neural network takes the flow signal time-series $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ as input and passes it through a sequence of **convolution layers** to generate m abstract feature vectors $Z = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_m)$ that are fed to a recurrent neural network (described in the next subsection) for further processing.

We use a variation of ResNet architecture [17] as our base convolution neural network. Figure 3 shows the network. Each convolution layer involves a **1D convolution** operation followed by a rectified linear unit (ReLU) non-linear activation [26], a dropout, and (optionally) a max-pooling operation. Let $\mathcal{X} \in \mathbb{R}^T$ be the input sequence of length m to a convolution layer (at any depth) with j-th kernel K^j of size W and stride size s. The $1D$ convolution operation at $t \in \{1, 2, \cdots, T\}$ is defined as:

$$\Phi_t \quad = \quad f(\sum_{i=1}^{W} K_i^j \mathcal{X}_{t+i.s-1} + b) \tag{1}$$

where b is the kernel K's bias and $f(\cdot)$ represents the activation function ReLU defined as $f(z) = max(z, 0)$. The outputs of convolution at each t are concatenated to produce a feature map $\Phi^j = [\Phi_1, \cdots, \Phi_{(T-W)/s+1}]$. With N such kernels, we get N feature maps represented as $\Phi \in \mathbb{R}^{NXO}$, where $O = (T - W)/s + 1$.

Convolution operations help extract the local features of time-series signal in a location invariant way. Strides s and filter width W capture the transitions in the input and receptive field or catchment of the convolution operation, respectively. We utilize **dropout** on the rectified activations to avoid overfitting [27]. For some layers, the convolution-pooling operations are succeeded by a **max-pooling** operation.

Additionally, we use **residual connections** between two layers so that a new layer added to the network learns something new. They also help with the diminishing gradient of preceding layers problem in deep convolution networks by forcing the network to learn the identity mapping [28]. Formally, let $\mathcal{X}$ be the input to a convolution layer, and $\mathcal{F}(\mathcal{X})$ represent the output of repeated convolution-pooling layers. The residual connection is defined as:

$$\mathcal{X}'' = \mathcal{F}(\mathcal{X}) + U^{\mathbf{T}}\mathcal{X} \tag{2}$$

where U is a transformation matrix that is used to bring $\mathcal{X}$ to the same dimensions as of $\mathcal{F}(\mathcal{X})$. In case both have the same dimensions, U becomes an identity matrix. Residual connections are usually used after one or two convolution-pooling layers. We use it between second and fourth layers in our network (Figure 3).

As shown in Figure 3, the first convolution layer in our network uses 256 different kernels (*i.e.,* output channels) each of size 10 and stride 2. This generates 432,000 feature values in each feature map. The second convolution layer then applies a kernel of window size 10 and stride 2 to each feature map and reduces the number of features in each feature map to 43,200. This process continues until the last convolution layer, which generates $m = 900$ features in each feature map. In other words, the output of the CNN is $Z \in \mathbb{R}^{256X900}$.

C. Recurrent Neural Network

Recurrent neural networks are used to model inputs with sequential nature. Since our data is a time-series sequence, we use the recurrent layer to model the temporal nature of our signal that builds on the high-level features from the convolution layers. The **recurrent** layer takes the feature vectors $Z = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_m)$ produced by the preceding ResNet CNN as input, and computes a representation $\mathbf{h}_t$ at every time step t by combining the current input $\mathbf{z}_t$ with the output of the previous time step $\mathbf{h}_{t-1}$. The recurrent units model the temporal dynamics of the input signal, by working on the sharp feature maps of the CNN.

We use Gated Recurrent Units (GRUs) [29] as our recurrent units. GRU has two gates: update gate (u) and reset gate (r) apart from the hidden cell state h. It combines the forget gate and the input gate of the popular Long Short-Term Memory

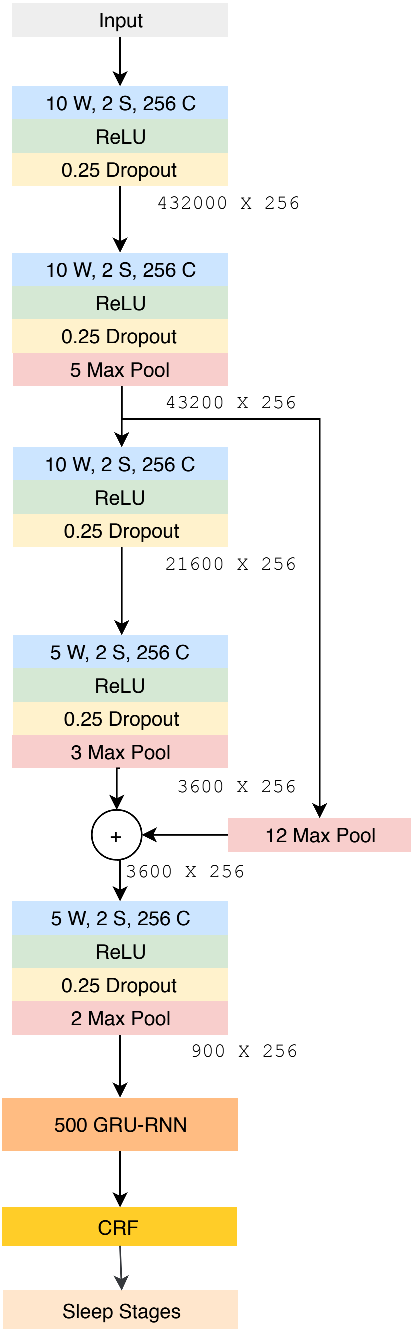


Fig. 3: Our deep learning model architecture, with CNN layers, GRU connections, and CRF for end-to-end learning. W refers to the Kernel size, S refers to the stride size, and C refers to the channel size at each convolution layer. The numbers indicated at the end of each convolution layer represent the size of output at each layer.

(LSTM) [30] unit into one update gate. The update equations for GRU can be written as:

$$\mathbf{u}_t = \sigma(W_z \mathbf{z}_t + U_z \mathbf{h}_{t-1} + b_z) \quad (3)$$

$$\mathbf{r}_t = \sigma(W_r \mathbf{z}_t + U_r \mathbf{h}_{t-1} + b_r) \quad (4)$$

$$\widetilde{\mathbf{h}}_t = 2\sigma(W_h \mathbf{z}_t + \mathbf{r}_t \odot U_h \mathbf{h}_{t-1} + b_h) - 1 \quad (5)$$

$$\mathbf{h}_t = \mathbf{u}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{u}_t) \odot \widetilde{\mathbf{h}}_t \quad (6)$$

where, $\sigma(\cdot)$ is the sigmoid activation function defined as $\sigma(x) = 1/(1+e^{-x})$, W's and U's are weight matrices, b's are biases, and $\odot$ denotes the Hadamard or element-wise product. GRUs have been shown to be much faster owing to reduced number of parameters and perform at par with LSTMs [31], [32].

The vector $\mathbf{h}_t$ effectively represents each 30-second epoch in context, which can be used to classify the epoch into one of the sleep stages using a softmax layer. Formally, the probability of k-th class for classifying into K sleep stages is

$$p(y_t = k|\mathbf{h}_t, W_o) = \frac{\exp\ (W_{o,k}\mathbf{h}_t + b_k)}{\sum_{k=1}^{K} \exp\ (W_{o,k}\mathbf{h}_t + b_k)} \quad (7)$$

where W are the classifier weights, and b are the bias terms. We minimize the negative log likelihood (NLL) of the gold labels. The NLL for one data point $(\mathbf{x}, \mathbf{y})$ is:

$$\mathcal{L}_c(\theta) = -\sum_{k=1}^{K}\sum_{t=1}^{m} \mathcal{I}(y_t = k) \log p(y_t = k|\mathbf{x}, \theta) \quad (8)$$

where θ denotes the set of model parameters, and $\mathcal{I}(y = k)$ is an indicator function to encode the gold labels: $\mathcal{I}(y = k) = 1$ if the gold label $y = k$, otherwise 0.[1] The loss function minimizes the cross-entropy between the predicted distribution and the target distribution (*i.e.,* gold labels). We refer to this combined architecture (*i.e.,* an RNN layer on top of a CNN) as **R-CNN**.

D. Conditional Random Field

1) ***Motivation****:* The R-CNN model presented above works in the input signal space and predicts the sleep stage for each time step independently based on the corresponding RNN hidden state. Although it considers the input context through recurrent layers, it is oblivious to the dynamics in the output space, *i.e.,* dynamics of the sleep stages.

Prior works using deep neural networks have focused entirely on extracting the best features from the input signals like EOG, ECG, or RF signals for predicting the sleep stage [14], [15], [9], [6], [11], [13]. Like R-CNN, these methods make independent (as opposed to collective) decisions. We argue that this approach is not optimal especially when there are strong dependencies across output labels. It is known that the sleep stage transitions have a strong dependency structure [16]. For example, a number of transitions are not allowed as can be seen in Figure 2. Also, there could be complex dependencies like the long-term cyclical effect of events like arousal or K-complex spindles on deep and REM sleep states [25]. Exploiting this transition structure for an accurate sleep staging is important. Also, because of local normalization (*i.e.,* softmax in Equation 7), these models suffer from the so-called label bias problem [33].

Instead of modeling classification decisions independently, we model them jointly using a conditional random field or **CRF** [33]. In our network, we put the CRF layer above the recurrent layer of R-CNN, and train the whole network end-to-end. CRFs have been shown to be able to utilize the global temporal context for maximizing sequence probabilities, relying upon first- or higher-order Markov assumptions over the output label transitions.

2) ***Neural CRF****:* The input to our CRF layer is a sequence of hidden states $H = (\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_m)$ from the GRU-based recurrent layer, and the corresponding label sequence is $\mathbf{y} = (y_1, y_2, \cdots, y_m)$. The compatibility of the input feature H and an output label $y_t \in \{A, R, D, L\}$ at time step t is computed by the unary (node) potential defined as:

$$\mathbf{\Psi}_n(y_t|H, \mathbf{w}_n, b_n) = \mathsf{exp}(\mathbf{w}_n^T \phi(y_t, H) + b_n) \quad (9)$$

where $\phi(\cdot)$ denotes the feature vector computed from the input and the sleep stage labels, and $\mathbf{w}_n$ is the associated weight vector. Here, $\mathbf{\Psi}_n(y_t|H, \mathbf{w}_n, b_n)$ can be considered as a score (unnormalized probability) given to label y_t. Applying the node potential to all nodes in the sequence generates a matrix S of size $m \times |K|$, where K is the set of sleep stages/classes (in our case $|K| = 4$), and $S_{i,j}$ corresponds to the score of the j-th class for input $\mathbf{h}_i$.

To model dynamics in the label sequence, we define edge potentials between y_{t-1} and y_t as:

$$\mathbf{\Psi}_e(y_{t-1}, y_t|H, \mathbf{w}_e, b_e) = \mathsf{exp}(\mathbf{w}_e^T \phi(y_{t-1}, y_t, H) + b_e) \quad (10)$$

where $\phi(y_{t-1}, y_t, H)$ denotes edge features with $\mathbf{w}_e$ being the corresponding weight vector. The edge potential computes a score for each possible edge transition in a matrix of size $|K| \times |K|$. The joint conditional probability for the sequence is defined as:

$$p(\mathbf{y}|H, \theta) = \frac{1}{Z(H, \theta)} \prod_{t=1}^{m} \mathbf{\Psi}_n(y_t|H, \mathbf{w}_n, b_n) \prod_{t=2}^{m} \mathbf{\Psi}_e(y_{t-1}, y_t|H, \mathbf{w}_e, b_e) \quad (11)$$

where $Z(H, \mathbf{w}_n, \theta)$ is the global normalization constant (partition function) derived as the sum over all possible sequences and θ denotes the set of all parameters in the complete (R-CNN-CRF) network.

$$Z(H, \theta) = \sum_{\mathbf{y}} \prod_{t=1}^{m} \mathbf{\Psi}_n(y_t|H, \mathbf{w}_n, b_n) \prod_{t=2}^{m} \mathbf{\Psi}_e(y_{t-1}, y_t|H, \mathbf{w}_e, b_e) \quad (12)$$

[1] This is also known as one-hot vector representation.

This global normalization constrains the distribution to a valid probability distribution and helps overcome the label bias problem of locally normalized models. The negative log-likelihood for one data point can be expressed as

$$\mathcal{L}(\theta) = -\log p(\mathbf{y}|H,\theta) \tag{13}$$

$$= \log \mathrm{Z} - \sum_{t=1}^{m} \mathbf{w}_n^T \phi(y_t, H) - b_n$$

$$- \sum_{t=2}^{m} \mathbf{w}_e^T \phi(y_{t-1}, y_t, H) - b_e \tag{14}$$

Note that the objective in Equation 14 is convex with respect to the CRF parameters $\theta' = \{\mathbf{w}_e, \mathbf{w}_n, b_e, b_n\}$ assuming the inputs from the R-CNN (*i.e.,* Z) are fixed. In training, we add a l_1 regularization on the CRF parameters θ' to promote sparsity. The final objective can thus be written as

$$\min_{\theta} \mathcal{L}(\theta) + \lambda \left\|\theta'\right\|_1 \tag{15}$$

As can be seen in Figure 2, a number of transitions are not observed while some have a large value making l_1 norm more appropriate to our problem compared to a l_2 norm.

The complete R-CNN-CRF network is trained end-to-end on the loss in Equation 15 by back-propagating the errors (gradients) to the R-CNN. Similar end-to-end training has shown impressive results in computer vision [34]. Once the parameters of the network are learned, decoding the most probable sequence can be performed effectively using the Viterbi algorithm.

E. Class Distribution Cost-Sensitive Prior

After analyzing our training data, we found that the class distribution of different sleep stages is skewed with REM and Deep sleep forming less than 10% of annotations each. To tackle this issue, we add a class prior α_k over the log likelihood. The class prior α_k for $k \in \{1, 2, \ldots, K\}$ is estimated from the training data by:

$$\alpha_k = \frac{n_\mu}{n_k} \tag{16}$$

where n_μ is the average number of labels in each class *i.e.,* n/K with n being the total number of sleep labels in the training set. We incorporate the class prior in our loss from Equation 15 as:

$$\min_{\theta} \; - \sum_{k=1}^{K} \sum_{t=1}^{m} \mathcal{I}(y_t = k) \alpha_k \log \; p(y_t = k|\theta) + \lambda \left\|\theta'\right\|_1 \tag{17}$$

where $\mathcal{I}(.)$ is the boolean indicator function as defined before. The priors α being inversely proportional to the number of samples of the class giving more weight-age to the under-represented classes leading to balanced learning during the training phase. We demonstrate the benefit of this prior in our experiments.

IV. Experiments

In this section, we describe our dataset, the metrics used, and baseline methods we compare our approach against.

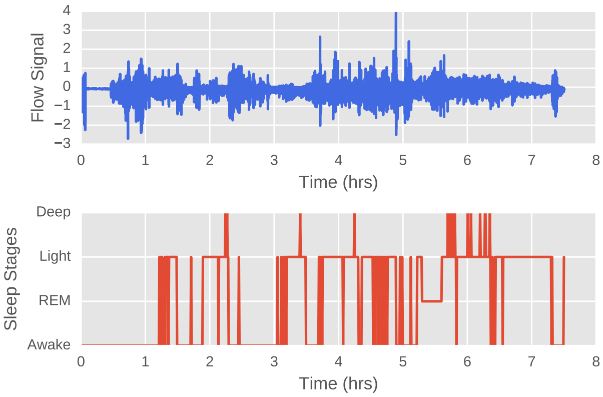


Fig. 4: An illustrative example of flow signal and corresponding sleep stage annotations for a subject.

A. Dataset

We use the publicly available data from Multi-Ethnic Study of Atherosclerosis (MESA) [35] provided by National Sleep Research Resource (NSRR) as part of an initiative to provide the health informatics community for building tools that can be helpful for sleep research and health-care [36]. We use the nasal airflow pressure channel data (univariate time-series), hereto referred to as flow signal, from PSG data of 400 randomly sampled sleep apnea patients.

The flow signal is sampled at 32 Hz (*i.e.,* 32 samples per second), while the sleep stages are annotated for contiguous 30-second epochs during the duration of the recording. The dataset comes with five sleep stages annotated: awake, REM, N1, N2, and N3/4. At the time of the MESA cohort collection, stages N3 and N4 were distinct entities. However, subsequent research demonstrated lack of fundamental differences between N3 and N4, and N4 has been subsumed into N3, otherwise known as deep sleep [1]. Prior studies have generally combined stages N1 and N2 into a unified group, light sleep, due to transitory nature of N1 and difficulty in differentiating between the two states. We follow the same approach, and we identify four states: awake, REM, light, and deep sleep. We take 7.5 hours of flow data for each patient for our experiments. An example of a single subject data including flow signal and sleep staging is shown in Figure 4.

B. Baseline Methods

We use the vanilla linear-chain CRF[33] and the R-CNN models as our baseline methods to compare against our model.

(a) **Conditional Random Field (CRF)**: We provide power spectral density features with the high-frequency band of 16 Hz as input to the CRF. We train it with Limited-memory BFGS learning algorithm with elastic net regularization. We use this baseline to show the benefit of using the deep network as a feature extractor instead of using the handcrafted features.

(b) **R-CNN**: It has the same architecture as described in Subsections III-B - III-C, with a softmax as the output layer at each time step. R-CNN classifies each time step independently and serves as the baseline deep learning method.

(c) **Conditional Adversarial Resnet-LSTM**: We implemented and trained the conditional adversarial R-CNN (ResNet with LSTM-RNN) architecture [9] until the discriminator loss stabilizes to entropy. Their network was shown to be able to remove the environmental noise from the representations learned by the network with state-of-the-art results using *Radiofrequency (RF)* based signals.

(d) **R-CNN with attention:** We added attention mechanism on top of our base R-CNN model. We used soft dynamic attention [37] in our model with a local window.

C. Variants of Our Neural CRF Model

We experiment with different variants of our neural CRF model.

(a) **Neural CRF**: This is the base R-CNN network augmented with linear-chain CRF, and trained end-to-end according to Eq. 14.

(b) **Second Order Neural CRF**: It builds up on the Neural CRF with second order edges of the form (y_t, y_{t+2}) in addition to the first order edges (y_t, y_{t+1}), thus captures longer dependencies.

(c) **Cost-sensitive Neural CRF**: It takes into consideration the class distribution priors from the training dataset according to Eq. 17 without using the l_1 regularization.

(d) **Regularized cost sensitive Neural CRF**: This regularizes the cost-sensitive Neural CRF with l_1 regularization (Eq. 17).

D. Evaluation Metrics

Cohen's Kappa coefficient (κ) and accuracy are the two commonly used metrics to compare a sleep staging model's predictions with ground truth annotations from PSG. In addition, we report the mean absolute error (MAE) of sleep efficiency.

(a) **Accuracy**: Accuracy is defined as a fraction of correct labels predicted by the model out of total number of annotations.

(b) **Kappa**: Cohen's Kappa coefficient (κ) [38] is a commonly used metric for sleep stage prediction quality that accounts for blind luck in model prediction. It measures the degree of concordance between two independent raters - model prediction and ground truth distribution. κ has values ranging from 0 to 1, with 0 being agreement by pure luck and 1 being a total convergence between the raters. A κ score of $<$0.4 is considered low, $>$0.4 moderate, $>$0.6 high, and $>$0.8 to be near perfect agreement with observed data. The sleep staging ground truth annotations by two or more human experts has a κ of **0.85** on our dataset. Thus, we can be very confident of the sleep stage annotations provided by MESA.

(c) **Mean Absolute Error (MAE) of Sleep Efficiency**: Sleep efficiency is a common metric used by sleep researchers for assessing sleep quality. It is defined as a fraction of time with non-wake sleep over the total duration of sleep. One of the uses of sleep staging is to calculate the sleep efficiency metric as a first pass metric over the quality of sleep. Sleep efficiency is given by:

$$\mathrm{SE} = \frac{n_R + n_L + n_D}{n_A + n_R + n_L + n_D} \tag{18}$$

where, n_A,n_R,n_L,n_D refer to number of 30-second epochs spent in awake and REM, light, and deep sleep, respectively. We calculate the absolute error of sleep efficiency for a patient $p \in \mathcal{P}$ as the absolute difference between estimated sleep efficiency ($\widehat{SE_p}$) from the predicted sleep stage labels and real sleep efficiency (SE_p). The MAE over the test set $\mathcal{P}$ is defined as:

$$\mathrm{MAE} = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \frac{|\widehat{SE_p} - SE_p|}{SE_p} \tag{19}$$

While we do not train our models to optimize MAE, we assess their performance with MAE. Accurate calculation of sleep efficiency is one of the applications of sleep staging that is very useful for both the patient and medical practitioner to monitor progress on CPAP therapy.

E. Hyper-parameter Tuning

We split our dataset based on subjects into 60% for training, 20% for validation, and 20% for testing, *i.e.,* we never train on data from a subject in the test dataset. We tune the hyperparameters on the validation set by using *early stopping* with parameters that provide the best validation κ score. The parameters chosen for the R-CNN model are shown in Figure 3. We experimented with different choices of CNN layers — kernel sizes (5, 10, 15, 20), strides (2, 3, 4, 5), channel sizes (128, 256, 512), and residual connections (1-to-3, 2-to-4, ResNeXt [39]) for our 5-layered CNN. For RNN, we experimented with hidden dimensions of 125, 250, 300, 500, and 750. We chose the regularization coefficients λ to be 0.005 after tuning on $\lambda \in \{0.01, 0.005, 0.0001\}$. Our implementation uses TensorFlow [40].

V. Results and Analysis

In this section, we present our findings on sleep stage classification and compare with the baselines and existing methods. We also present an analysis of our findings with the effect of cost-sensitive prior and saliency map visualizations of model predictions.

A. Classification and Sleep Efficiency Results

We categorize our methods into three: (*i*) linear (non-deep) CRF, (*ii*) deep neural models with the softmax output layer, and (*iii*) our deep neural models with CRF output layer. The results are shown in Table I. We summarize our findings below. (*i*) **Linear (non-deep) CRF:** As we can observe, the baseline CRF performs poorly with a low accuracy of 52.4%, κ of only 0.27, and higher MAE of 29.4 on sleep efficiency. It can be attributed to the low representational power of the input features compared with the task-specific feature extraction of deep learning architectures.

Approach	Accuracy	κ	SE MAE%
CRF	52.4%	0.28	29.4%
R-CNN	71.5%	0.49	12.5%
Conditional Adversarial [9]	71.1%	0.49	12.6%
Attentional R-CNN	70.7%	0.48	12.8%
Neural CRF	72.3%	0.54	10.9%
Neural CRF (Order 2)	72.5%	0.55	10.8%
Cost Sensitive Neural CRF	73.9%	0.56	10.3%
Regularized Cost Sensitive Neural CRF	**74.1%**	**0.57**	**9.9%**

TABLE I: Accuracy, Kappa, and Sleep efficiency (SE) MAE (lower the better) scores for different approaches.

(*ii*) **Deep neural models with softmax output:** The deep neural models improve the performance considerably over the non-deep CRF model by taking the accuracy to 71%, κ to 0.49, and MAE down to 12.5. Our baseline R-CNN with ResNet-GRU architecture performs the best overall. The conditional adversarial network [9] performs at par with the R-CNN, while it poses additional training challenges because of the instability caused by adversarial training [41]. Using the local attentional mechanism [37] leads to a slight drop in performance, possibly due to the extra parameters. The residual connection described in Section III-B was quite beneficial for the R-CNN model; it increased the κ scores from 0.29 to 0.49.

Remark: Our experiments that used LSTM units and bi-directional GRU/LSTM as recurrent units in R-CNN did not make a difference in the performance. To train faster, we use unidirectional GRUs in our experiments.

(*iii*) **Neural CRF models:** Adding the CRF layer to the base R-CNN improves the performance substantially taking the κ score to 0.54, a 10.2% increase in relative terms. Adding second order edges in the CRF marginally improves the performance, though it increases the run-time of the model substantially. Using a cost-sensitive version of the Neural CRF increases the performance considerably by 2% in κ over the Neural CRF, while the regularized cost sensitive CRF improves the performance by 4% over the Neural CRF model. Using CRF that infers the global temporal context improves the performance substantially. Adding domain dependent prior knowledge like cost-sensitive prior and sparse regularization helps bring additional gains in the model performance. While the increase in *accuracy may seem marginal*, the increase in κ **of 14% is substantial** since it reflects an improved detection of the difficult and less frequent deep and REM sleep stages.

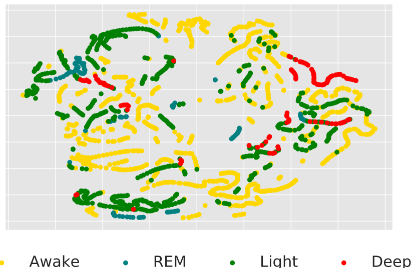


Fig. 6: t-SNE plot of GRU-RNN output from Regularized Cost-Sensitive Neural CRF for each annotation. RNN is able to differentiate the awake and light states to a good measure.

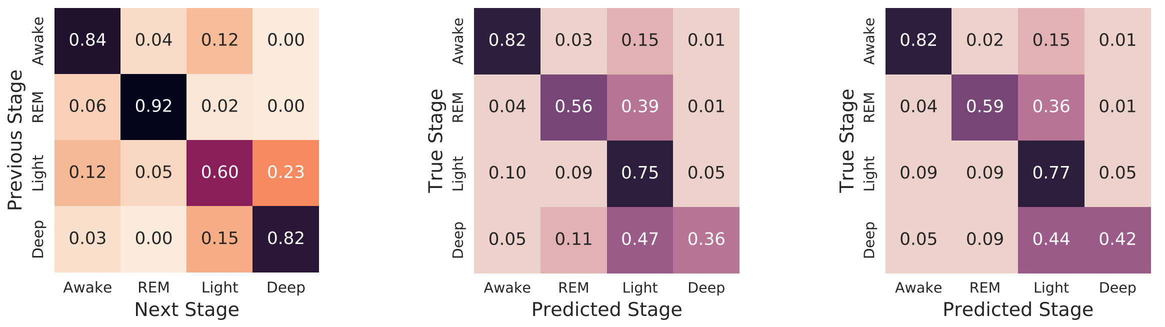
Transition matrix of the regularized Neural CRF model is shown in Figure 5a. As we can see the values in the matrix assign zero scores to the non-existent transitions in the Figure 2. We also found that these values are close to the ground truth transition values we observe in the dataset.

The deep non-linear layers help the model to extract the feature space that is very relevant to the task as reflected by the difference in performance of the non-deep CRF vs. R-CNN. Combining the two as our Neural CRF does is very helpful in leveraging the strengths of both approaches – deep learning for meaningful feature extraction in the input side, and the modeling strength of CRFs, which use global inference to model consistency in the output structure.

The precisely similar trend as above is observed with sleep efficiency MAE. Our models can predict sleep efficiency metric with a reasonable accuracy — within 10%-15% of the sleep efficiency value. This is expected since the model is able to differentiate the awake state with very high accuracy as shown by the confusion matrices in Figures 5b and 5c. Hence, our model can provide an accurate estimation of sleep efficiency to help health-care professionals track the response of CPAP therapy.

B. *Effect of Cost-Sensitive Training*

Since our data has under-represented REM and Deep sleep annotations, we used a cost-sensitive prior. We demonstrate the effect of this prior by showing the confusion matrices for

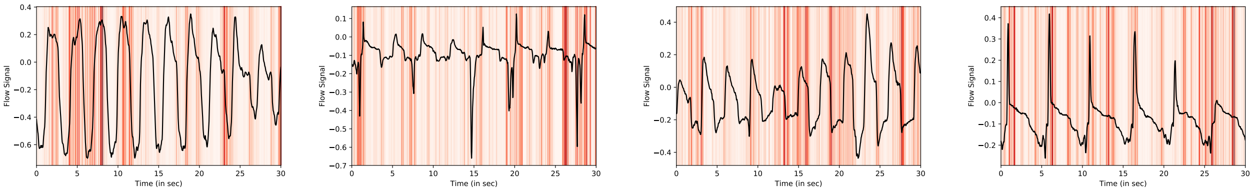


(a) CRF sleep stage transition matrix

(b) Neural CRF model

(c) Cost-Sensitive Neural CRF model

Fig. 5: Transition matrix from CRF of Regularized Cost-Sensitive Neural CRF from training (left). Confusion matrices of prediction from the baseline Neural CRF model (center) and cost-sensitive Neural CRF model (right) on the test dataset.



(a) Awake (b) REM (c) Light (d) Deep

Fig. 7: Saliency maps for the sample 30-second epochs correctly predicted by the model — one each for awake, REM, light and deep sleep stages. Darker shade reflects higher weighting by the model.

the Neural CRF and the cost-sensitive Neural CRF models in Figures 5b-5c. By adding cost sensitive prior, the class accuracies increase across the board, significantly for the under-represented classes of REM and Deep sleep. Hence, using a cost-sensitive prior for lifting the weights of the under-represented classes during training is helpful for reducing the effect of imbalanced class distributions.

Another point we would like to make is that our models are able to detect the awake state accurately and light sleep with good accuracy, but have real difficulty detecting the REM and deep sleep. This is also visible in the *t-SNE* [42] plot in Figure 6, where we plot in 2D the activations of the GRU-RNN layer of our cost sensitive neural CRF for a representative sample of annotations.

Note that the other signals such as no-contact and chest band based signals [9] have relatively lower performance on the awake and deep sleep states detection while are able to detect the REM and light sleep states accurately. However, making a direct comparison with these studies might be misplaced since they have used a young or healthy population of subjects, while our population consists of sleep apnea patients. Previous attempts [43] on sleep apnea patients have observed lower accuracy compared to the healthy subjects since the sleep dynamics exhibited by sleep apnea patients are harder to predict than those of healthy subjects.

C. Flow Signal Saliency

One of the most common criticisms of deep learning methods comes from the black-box nature of the models. We present the flow signal saliency as an exercise to interpret the model's basis for prediction. In our case, the CRF layer's transition matrix (Figure 5a) helped us understand the output sleep stage dynamics learned by the model. In order to get an understanding of how the model is predicting the sleep stages from the input flow signal, we adopt the saliency map approach proposed by Simonayan et al. [44] to interpret CNNs for image classification. The idea is to take the gradient of the classification scores with respect to the input image to learn weights of pixels the model is *"looking" at* while making predictions.

We use the same approach in our setting by learning the weights of model saliency over flow signal time-series by taking gradients of the classification scores with respect to the input flow signal. Figure 7 shows one representative sample 30-second epoch (that was correctly classified by model) for each sleep stage, and their saliency weights over the time-series values.

Through a visual inspection we can observe that the models seem to focus on two phases of the respiratory cycle, namely plateaus in flow closest to zero between inhalation and exhalation and periods of maximal change in flow rates. Respiratory rate variability [45] and respiratory effort amplitude [46] differ depending on stage of sleep. Thus it is conceivable that the models may be extracting information that approximates respiratory physiology features in trying to classify sleep stage. On the other hand, the model's saliency map may also represent new and unknown phenomena that could be useful for medical researchers to investigate.

VI. Conclusions

In this work, we present the first study on using flow signal for automated sleep staging. We utilize a neural CRF architecture that combines the representational power of deep neural networks with the modeling strength of structured output models to get the best of both worlds. For our neural model, we employ a deep CNN to learn high-level informative features from flow signals. A GRU-based RNN is used to encode features for classification by modeling temporal contextual information. The CRF jointly models the output sequence to capture temporal dynamics in the sleep states. Domain-dependent priors were used to regularize the network.

Our method substantially improves the classification performance over the baseline deep learning methods. We further demonstrate that using cost-sensitive prior for tackling class imbalance and sparse regularization on weights further improves the model performance. Our neural model (R-CNN) and the CRF approach can be augmented to the existing methods for improved sleep staging. In terms of implications for the sleep care, our method has an existing and immediate use case as it can be employed to track the response of patients to the CPAP therapy by automatically and accurately tracking sleep stages and overall sleep quality. Hence, our study is helpful in advancing clinical sleep research and motivates researchers to investigate the effect of CPAP on sleep architecture of subjects.

References

[1] M. H. Silber, S. Ancoli-Israel, M. H. Bonnet, S. Chokroverty, M. M. Grigg-Damberger, M. Hirshkowitz, S. Kapen, S. A. Keenan, M. H. Kryger, T. Penzel *et al.*, "The visual scoring of sleep in adults," *Journal of Clinical Sleep Medicine*, vol. 3, no. 02, pp. 22–22, 2007.

[2] A. V. Shelgikar, J. S. Durmer, K. E. Joynt, E. J. Olson, H. Riney, and P. Valentine, "Multidisciplinary sleep centers: strategies to improve care of sleep disorders patients," *Journal of clinical sleep medicine: JCSM: official publication of the American Academy of Sleep Medicine*, vol. 10, no. 6, p. 693, 2014.

[3] P. E. Peppard, T. Young, J. H. Barnet, M. Palta, E. W. Hagen, and K. M. Hla, "Increased prevalence of sleep-disordered breathing in adults," *American journal of epidemiology*, vol. 177, no. 9, pp. 1006–1014, 2013.

[4] N. F. Watson, "Health care savings: the economic value of diagnostic and therapeutic care for obstructive sleep apnea," *Journal of clinical sleep medicine: JCSM: official publication of the American Academy of Sleep Medicine*, vol. 12, no. 8, p. 1075, 2016.

[5] T. Lajnef, S. Chaibi, P. Ruby, P.-E. Aguera, J.-B. Eichenlaub, M. Samet, A. Kachouri, and K. Jerbi, "Learning machines and sleeping brains: automatic sleep stage classification using decision-tree multi-class support vector machines," *Journal of neuroscience methods*, vol. 250, pp. 94–105, 2015.

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

[7] J. Mantua, N. Gravel, and R. Spencer, "Reliability of sleep measures from four personal health monitoring devices compared to research-based actigraphy and polysomnography," *Sensors*, vol. 16, no. 5, p. 646, 2016.

[8] S. J. Redmond, P. de Chazal, C. O'Brien, S. Ryan, W. T. McNicholas, and C. Heneghan, "Sleep staging using cardiorespiratory signals," *Somnologie-Schlafforschung und Schlafmedizin*, vol. 11, no. 4, pp. 245–256, 2007.

[9] M. Zhao, S. Yue, D. Katabi, T. S. Jaakkola, and M. T. Bianchi, "Learning sleep stages from radio signals: A conditional adversarial architecture," in *International Conference on Machine Learning*, 2017, pp. 4100–4109.

[10] A. Tataraidze, L. Korostovtseva, L. Anishchenko, M. Bochkarev, and Y. Sviryaev, "Sleep architecture measurement based on cardiorespiratory parameters," in *Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the*. IEEE, 2016, pp. 3478–3481.

[11] O. Tsinalis, "Deep learning for automated sleep monitoring," 2016.

[12] S. Chambon, M. Galtier, P. Arnal, G. Wainrib, and A. Gramfort, "A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series," *arXiv preprint arXiv:1707.03321*, 2017.

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

[14] S. Biswal, J. Kulas, H. Sun, B. Goparaju, M. B. Westover, M. T. Bianchi, and J. Sun, "Sleepnet: Automated sleep staging system via deep learning," *arXiv preprint arXiv:1707.08262*, 2017.

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

[16] J. Kim, J.-S. Lee, P. Robinson, and D.-U. Jeong, "Markov analysis of sleep dynamics," *Physical review letters*, vol. 102, no. 17, p. 178104, 2009.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] R. S. Rosenberg and S. Van Hout, "The american academy of sleep medicine inter-scorer reliability program: sleep stage scoring," *Journal of clinical sleep medicine: JCSM: official publication of the American Academy of Sleep Medicine*, vol. 9, no. 1, p. 81, 2013.

[19] F. Ebrahimi, M. Mikaeili, E. Estrada, and H. Nazeran, "Automatic sleep stage classification based on eeg signals by using neural networks and wavelet packet coefficients," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*. IEEE, 2008, pp. 1151–1154.

[20] X. Long, J. Yang, T. Weysen, R. Haakma, J. Foussier, P. Fonseca, and R. M. Aarts, "Measuring dissimilarity between respiratory effort signals based on uniform scaling for sleep staging," *Physiological measurement*, vol. 35, no. 12, p. 2529, 2014.

[21] J. Hedner, G. Pillar, S. D. Pittman, D. Zou, L. Grote, and D. P. White, "A novel adaptive wrist actigraphy algorithm for sleep-wake assessment in sleep apnea patients," *Sleep*, vol. 27, no. 8, pp. 1560–1566, 2004.

[22] H. E. Montgomery-Downs, S. P. Insana, and J. A. Bond, "Movement toward a novel activity monitoring device," *Sleep and Breathing*, vol. 16, no. 3, pp. 913–917, 2012.

[23] W. Gu, Z. Yang, L. Shangguan, W. Sun, K. Jin, and Y. Liu, "Intelligent sleep stage mining service with smartphones," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2014, pp. 649–660.

[24] A. Gautam, V. S. Naik, A. Gupta, S. Sharma, and K. Sriram, "An smartphone-based algorithm to measure and model quantity of sleep," in *Communication Systems and Networks (COMSNETS), 2015 7th International Conference on*. IEEE, 2015, pp. 1–6.

[25] L. De Gennaro and M. Ferrara, "Sleep spindles: an overview," *Sleep medicine reviews*, vol. 7, no. 5, pp. 423–440, 2003.

[26] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 630–645.

[29] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

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

[31] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *International Conference on Machine Learning*, 2015, pp. 2342–2350.

[32] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[33] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[34] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1529–1537.

[35] D. E. Bild, D. A. Bluemke, G. L. Burke, R. Detrano, A. V. Diez Roux, A. R. Folsom, P. Greenland, D. R. JacobsJr, R. Kronmal, K. Liu *et al.*, "Multi-ethnic study of atherosclerosis: objectives and design," *American journal of epidemiology*, vol. 156, no. 9, pp. 871–881, 2002.

[36] D. A. Dean, A. L. Goldberger, R. Mueller, M. Kim, M. Rueschman, D. Mobley, S. S. Sahoo, C. P. Jayapandian, L. Cui, M. G. Morrical *et al.*, "Scaling up scientific discovery in sleep medicine: the national sleep research resource," *Sleep*, vol. 39, no. 5, pp. 1151–1164, 2016.

[37] H. Mei, M. Bansal, and M. R. Walter, "Coherent dialogue with attention-based language models," *CoRR*, vol. abs/1611.06997, 2016. [Online]. Available: http://arxiv.org/abs/1611.06997

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

[39] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 5987–5995.

[40] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[41] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *CoRR*, vol. abs/1701.07875, 2017.

[42] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[43] S. J. Redmond and C. Heneghan, "Cardiorespiratory-based sleep staging in subjects with obstructive sleep apnea," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 3, pp. 485–496, 2006.

[44] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[45] G. Gutierrez, J. Williams, G. A. Alrehaili, A. McLean, R. Pirouz, R. Amdur, V. Jain, J. Ahari, A. Bawa, and S. Kimbro, "Respiratory rate variability in sleeping adults without obstructive sleep apnea," *Physiological reports*, vol. 4, no. 17, 2016.

[46] N. Douglas, D. White, C. K. Pickett, J. Weil, and C. Zwillich, "Respiration during sleep in normal man." *Thorax*, vol. 37, no. 11, pp. 840–844, 1982.