DeepSleepNet-Lite: A Simplified Automatic Sleep Stage Scoring Model With Uncertainty Estimates

Luigi Fiorillo, *Member, IEEE*, Paolo Favaro, *Member, IEEE*, and Francesca Dalia Faraci, *Member, IEEE*

***Abstract*—Deep learning is widely used in the most recent automatic sleep scoring algorithms. Its popularity stems from its excellent performance and from its ability to process raw signals and to learn feature directly from the data. Most of the existing scoring algorithms exploit very computationally demanding architectures, due to their high number of training parameters, and process lengthy time sequences in input (up to 12 minutes). Only few of these architectures provide an estimate of the model uncertainty. In this study we propose DeepSleepNet-Lite, a simplified and lightweight scoring architecture, processing only 90-seconds EEG input sequences. We exploit, for the first time in sleep scoring, the *Monte Carlo dropout* technique to enhance the performance of the architecture and to also detect the uncertain instances. The evaluation is performed on a single-channel EEG Fpz-Cz from the open source Sleep-EDF expanded database. DeepSleepNet-Lite achieves slightly lower performance, if not on par, compared to the existing state-of-the-art architectures, in overall accuracy, macro F1-score and Cohen's kappa (on Sleep-EDF v1-2013 ±30mins: 84.0%, 78.0%, 0.78; on Sleep-EDF v2-2018 ±30mins: 80.3%, 75.2%, 0.73). *Monte Carlo dropout* enables the estimate of the uncertain predictions. By rejecting the uncertain instances, the model achieves higher performance on both versions of the database (on Sleep-EDF v1-2013 ±30mins: 86.1.0%, 79.6%, 0.81; on Sleep-EDF v2-2018 ±30mins: 82.3%, 76.7%, 0.76). Our lighter sleep scoring approach paves the way to the application of scoring algorithms for sleep analysis in real-time.**

***Index Terms*—Sleep scoring, deep learning, model uncertainty estimation.**

Manuscript received January 15, 2021; revised June 5, 2021 and September 14, 2021; accepted September 30, 2021. Date of publication October 14, 2021; date of current version October 15, 2021. *(Corresponding author: Luigi Fiorillo.)*

This work involved human subjects or animals in its research. The authors confirm that all human/animal subject research procedures and protocols are exempt from review board approval.

Luigi Fiorillo is with the Institute of Informatics, University of Bern, 3012 Bern, Switzerland, and also with the Department of Innovative Technologies, Institute of Digital Technologies for Personalized Healthcare (MeDiTech), University of Applied Sciences and Arts of Southern Switzerland, 6928 Lugano, Switzerland (e-mail: luigi.fiorillo@supsi.ch).

Paolo Favaro is with the Institute of Informatics, University of Bern, 3012 Bern, Switzerland.

Francesca Dalia Faraci is with the Department of Innovative Technologies, Institute of Digital Technologies for Personalized Healthcare (MeDiTech), University of Applied Sciences and Arts of Southern Switzerland, 6928 Lugano, Switzerland.

Digital Object Identifier 10.1109/TNSRE.2021.3117970

I. Introduction

GOOD sleep plays a crucial role in human well-being, and sleep disorders represent a significant and an increasing public health problem [1]. Polysomnography (PSG) is used in sleep medicine as a diagnostic tool, so as to objectively analyze the quality of sleep and the common sleep pathologies - e.g. sleep breathing disorders, narcolepsy, sleep-related movement disorders [2]. Electroencephalography (EEG), electrooculography (EOG), electromyography (EMG) and electrocardiography (ECG) signals are essential for the PSG exam. The physicians extract sleep cycle information through the well-known sleep stage scoring procedure, according to the AASM guidelines [3]. The whole-night sleep recording is divided into 30-second windows, called epochs, and each epoch is classified into one of the following five sleep stages: wakefulness W, stage N1, stage N2, stage N3, and stage R (REM sleep). This manual sleep stage classification is obviously time-consuming and is affected by human error - several works report high values of inter- and intra-scorer variabililty [5]. Since 1960, a wide variety of techniques have been devised in an effort to automate this procedure. Still, up to now, no system has completely replaced the physician.

In the last decades, deep learning algorithms have been widely used to solve the sleep scoring task automatically. A thorough review of the application of deep learning architectures to sleep scoring can be found in [6]. Autoencoders [7], deep neural networks (DNNs) [8], convolutional neural networks (CNNs) [9]–[16], recurrent neural networks (RNNs) [17] and several combination of them [18]–[26] have been recently proposed. The main advantage of all these networks is the ability to learn features directly from raw data, by taking into account the temporal dependency among the sleep stages. However, the architectures of these models are quite complex, a high number of parameters need to be trained. The most recent ones process lengthy time sequences in input - i.e. up to 12 minutes - using RNNs, thus requiring extra resources to buffer the PSG input and making them unsuitable in home-monitoring and in real-time applications. As a rule, deep architectures with a high number of layers and parameters need to be trained on large databases to prevent overfitting. In different scenarios sleep datasets have a limited number of labeled PSG samples available. Lighter architectures may be better suited if the model needs to be trained from scratch. We found only two architectures [14], [21] performing the automatic sleep scoring and also providing an estimate of the model uncertainty. In [14] they use an additional classification block-2 (i.e. multilayer perceptron in cascade to the deep convolutional scoring architecture) to output the final sleep stage and the associated relative confidence score. In contrast, [21] trains 16 different models and uses the relative model variance to estimate the uncertain predictions. It is important to know the level of confidence of each prediction, as it could be the key to identify the misclassified sleep stages.

In this paper, we propose DeepSleepNet-Lite, a simplified and lightweight automatic sleep scoring architecture. It provides the predicted sleep stages along with an estimate of their uncertainty. The major advantage is that it does not require any additional computation over the baseline architecture to provide the estimate.

The two main contributions of this paper are: 1) the optimization of a simple feed-forward sleep scoring architecture, that processes only 90-second single-channel EEG in input; 2) the application of the *Monte Carlo dropout* sampling technique, using *dropout* at test time to capture the model uncertainty and to enhance the performance of the scoring system. In Section II we describe the architecture, the training algorithm and the regularization techniques used in our scoring system. In Section III we briefly present the label smoothing technique used to calibrate the scoring architecture. Moreover we propose a new conditional probability distribution computed over the targets (i.e. our prior knowledge), and used to smooth our labels. In Section IV we present the *Monte Carlo dropout* sampling technique, and its application within our sleep scoring system to estimate the uncertainty of the model. In the last sections, we demonstrate the efficiency of label smoothing and *Monte Carlo dropout* techniques in both calibrating and enhancing the performance of our model. We also demonstrate the efficiency of the uncertainty estimate procedure, by showing that it is able to identify the most challenging sleep stage predictions. We finally show that DeepSleepNet-Lite achieves performance on par with most up-to-date scoring systems.

II. DeepSleepNet-Lite

The architecture of DeepSleepNet-Lite is strongly inspired by *DeepSleepNet* from Supratak [18]. Unlike the original network, we have employed only the first *representation learning* part, and trained it with a *sequence-to-epoch* learning approach. The architecture receives in input a sequence of PSG epochs, and predicts the corresponding target of the central epoch of the sequence. In [27] we had already shown that the first *representation learning* part of the architecture, trained with a small temporal context - 90-second epochs, does most of the work on a small-sized database.

A. The Architecture

The *representation learning* architecture consists of two parallel CNNs employing small (CNN_{θ_S}) and large (CNN_{θ_L}) filters at the first layer. The small filter has been used to extract high-time resolution patterns, while the large filter has been used to extract high-frequency resolution patterns. The idea behind the use of the small and large filter sizes

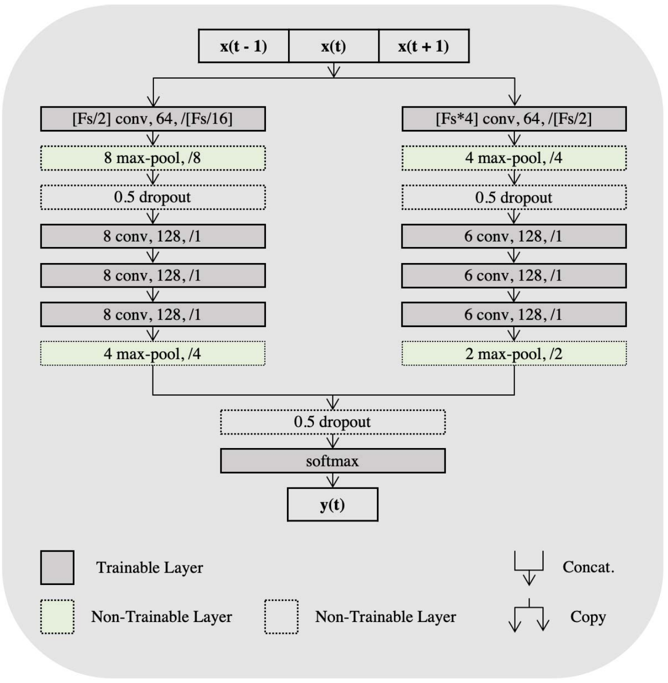

Fig. 1. An overview of the *representation learning* architecture from [18], with our *sequence-to-epoch* input-output training approach.

comes from the way the signal processing experts define the trade-off between temporal and frequency precision in the feature extraction procedure [28]. Each CNN section consists of four convolutional layers and two max-pooling layers. Each convolutional layer executes three operations: a one-dimensional convolution of the filters with the 90-second epochs, a batch normalization [29] and an element-wise rectified linear unit (ReLU) activation function. The filter size, the number of filters and the stride size of each *conv* layer are defined in Fig. 1. The pooling layer is used to downsample the input. In each *max-pool* unit the pooling size and the stride size are specified.

The 90-second EEG signal $\mathbf{x}_i$ is given in input to the convolutional neural networks CNN_{θ_S} and CNN_{θ_L}. The parameters θ of each convolutional neural network are independently trained, so as to return in output two feature vectors $\mathbf{h}_i{}^S$ and $\mathbf{h}_i{}^L$. The outputs are concatenated in $\mathbf{f_i}$, then forwarded to the *softmax* layer.

$$\mathbf{h}_i{}^S = CNN_{\theta_S}(\mathbf{x}_i) \tag{1}$$
$$\mathbf{h}_i{}^L = CNN_{\theta_L}(\mathbf{x}_i) \tag{2}$$
$$\mathbf{f}_i = \mathbf{h}_i{}^S || \mathbf{h}_i{}^L \tag{3}$$

The softmax function, together with the cross-entropy loss function, is used to train the model to output the logits $\mathbf{z}_i$ and the probability for the five mutually exclusive classes that correspond to the five sleep stages.

$$\mathbf{z}_i = \mathbf{W}^T\mathbf{f}_i + \mathbf{b} \tag{4}$$
$$\hat{p}_{i,k} = \frac{exp(z_{i,k})}{\Sigma_j exp(z_{i,j})} \tag{5}$$

where $\theta = \{\boldsymbol{W},\boldsymbol{b}\}$ are the parameters of the softmax layer, j is the index of the vector $\mathbf{z}$, $\hat{p}_{i,k}$ is the output probability of class k associated to x(t), the centred 30-second signal in $\mathbf{x}_i$.

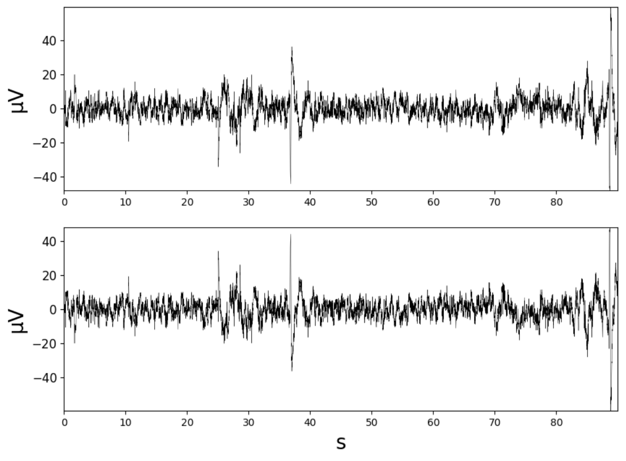

Fig. 2. *(top)* 90-s EEG raw signal. *(bottom)* 90-s EEG vertically flipped.

All the model specifications are reported in Fig. 1, equally to the first *representation learning* in [18].

B. Training Algorithm

The architecture is trained end-to-end via backpropagation, using the *sequence-to-epoch* learning approach. Classification algorithms learn to predict the most represented class in the training set, leading to the so called class imbalance problem. Here the least represented classes are balanced by using two techniques: *(i) data augmentation*, by flipping vertically the data input (i.e. multiply by -1 the original signal, see Fig. 2) belonging to the least represented classes, then *(ii) oversampling* randomly the data so that all the sleep stages are equal in number to the most represented class. In our model, the input is a sequence of three 30-second epochs, and the output is the corresponding target of the central epoch at time (t). So, we refer to the target of the central epoch to compute the most or least represented classes.

The model is trained using mini-batch Adam gradient-based optimizer [30] with a learning rate lr. The training procedure runs up to a maximum number of iterations, as long as the break early stopping condition is satisfied - further details in the next subsection II-C.

C. Regularization Techniques and Training Parameters

Dropout. Commonly used as regularizer in convolutional neural networks, it prevents overfitting and co-adaptation of the feature maps [31]. During the training procedure a certain number of neurons are randomly removed, dropping units with a probability p. We fix the probability of dropping a connection equal to 50%, i.e. $p = 0.5$.

Early stopping. It provides guidance on how many iterations can be run before the model begins to overfit [32]. The training procedure should be stopped as soon as the performance (i.e. F1-score) on the validation set is lower than it was in the previous iteration step. However, in this study, before hastily stopping the learning procedure, the algorithm runs for an additional number of iterations (by fixing the so called *patience* parameter). The model with the highest performance is the one we finally save.

L2 weight decay. This technique simply adds a term to the loss function that penalizes the weight values; by doing so it avoids the exploding gradient phenomena [33]. The $lambda$ defines the degree of penalty and it has been set to 10^{-3}.

All the training parameters are fixed as in [18]. The Adam optimizer's parameters $beta1$ and $beta2$ have been set to 0.9 and 0.999 respectively. The mini-batch size has been set to 100. During the batch normalization procedure, the ϵ value of 10^{-5} has been added to the mini-batch variance. In order to compute the mean and variance of the training samples, the moving average has been implemented using a fixed decay rate value of 0.999. The learning rates parameter lr has been fixed to 10^{-4}. The maximum number of iterations has been set to 100, with the early stopping *patience* parameter equal to 50.

III. Model Calibration

Along with the estimated sleep stage, the model should also provide a calibrated confidence - i.e. the probability associated to the predicted stage should mirror its ground truth correctness likelihood. We adopted label smoothing [34] to calibrate our model. It has been shown to be a suitable technique to improve model calibration [35].

In a standard training of a neural network, the cross-entropy loss is minimized using the hard targets y_{k} (i.e. hot encoded targets, '1' for the correct class and '0' for the other). For a network trained with label smoothing, the hard targets are weighted with the *uniform distribution* $1/K$ (eq. 6), and the cross-entropy loss is minimized using the weighted mixture of the targets (eq. 7).

$$y_{\mathrm{k}}{}^{LS_{\mathrm{u}}} = y_{\mathrm{k}} \cdot (1-\alpha) + \alpha/K \tag{6}$$

$$H(\boldsymbol{y},\boldsymbol{p}) = \sum_{k=1}^{K} -y_{\mathrm{k}}{}^{LS_{\mathrm{u}}} \cdot \log(\hat{p}_{\mathrm{k}}) \tag{7}$$

where α is the smoothing parameter, K is the total number of classes, $y_{\mathrm{k}}{}^{LS_{\mathrm{u}}}$ the targets smoothed with the *uniform distribution*, and $\hat{p}_{\mathrm{k}}$ the softmax output probabilities.

A. Conditional Probability Distribution in Label Smoothing

In our study, we introduce a new distribution to smooth the labels, by mainly taking into account the importance in sleep scoring of the transitions from one sleep stage to the other. The idea is to compute the *conditional probability distribution* over the five sleep stages of all the sequences of epochs:

$$\boldsymbol{M} = P(stage(t)|stage(t-1), stage(t+1)) \tag{8}$$

where in $\boldsymbol{M}$ we have the conditional probability values for each possible combination of sequences of three sleep stages. In detail, we compute the probability to be in a stage at time t given the previous $(t-1)$ and the next $(t+1)$ sleep stages over the whole database. The matrix $\boldsymbol{M}$ is $K \times K \times K$ dimensional, where K is the total number of sleep stages.

As stated previously, the architecture takes in input a sequence of three epochs, and outputs the corresponding target of the central epoch $y_{\mathrm{k,(t)}}$. So, during the training procedure,

TABLE I
CONDITIONAL PROBABILITY VALUES COMPUTED OVER THE SEQUENCES, EXTRACTED FROM THE SLEEP-EDF V1-2013 DATASET, WITH THE LABEL AT TIME $(t-1)$ FIXED IN AWAKE. *i.e.* $\boldsymbol{M}_{W,K\times K}$

W(t-1)	*W(t+1)*	*N1(t+1)*	*N2(t+1)*	*N3(t+1)*	*R(t+1)*
W(t)	0.991	0.503	0.131	0.333	0.217
N1(t)	0.008	0.495	0.581	0.000	0.109
N2(t)	0.000	0.002	0.275	0.000	0.000
N3(t)	0.000	0.000	0.006	0.667	0.000
R(t)	0.000	0.000	0.006	0.000	0.674

given the knowledge of the sleep stage at time $(t-1)$ and the sleep stage at time $(t+1)$, the hot encoded $y_{\mathrm{k,(t)}}$ will be smoothed with the corresponding conditional probability vector from $\mathbf{M}$.

In Table I we report an example of the conditional probability values computed over the sequences extracted from the Sleep-EDF v1-2013 dataset (see section V), with the label at time $(t-1)$ fixed in sleep stage awake. We highlight in light-green an example of the conditional probability vector to use when we had awake W at time $(t-1)$ and $N1$ at time $(t+1)$, which results in the following smoothed target:

$$y_{\mathrm{k}}{}^{LS_{\mathrm{s}}} = y_{\mathrm{k}} \cdot (1-\alpha) + \alpha \cdot \boldsymbol{M}_{W,K,N1} \tag{9}$$

The cross-entropy loss is minimized using the weighted mixture of the hard targets with these conditional probability distributions.

The smoothing parameter α for the *uniform distribution* and the *conditional probability distribution* weighting has been set to 0.1 and 0.2 respectively. These two values gave us the highest performance. In both, we explored α values up to 0.5.

IV. ESTIMATING UNCERTAINTY

In order to estimate the model uncertainty, we exploit the *dropout* regularization technique. As explained above, during the training procedure, at each iteration, *dropout* removes a certain number of units within our network at random. It randomly samples a certain number of sub-networks, so that each time the model's architecture is slightly different. In a standard application, *dropout* is used only during the training phase. At test time, instead, all the trained neurons and connections are used - i.e. all the weights of the whole network. The output could be interpreted as an averaging ensemble of all the sub-networks. We employ, for the first time in sleep staging, the *Monte Carlo* (*MC*) *dropout* [36], to quantify the model uncertainty, and to further enhance the performance of the scoring architecture. Monte Carlo refers to a specific class of algorithms that rely on random sampling, to provide estimates and distributions of numerical quantities. *MC dropout* simply consists in applying the randomized sampling even at test time. The different sub-networks could be interpreted as *Monte Carlo* samples extracted from the space of all the possible models. As a result, by applying *dropout* N times at inference time (with the probability of dropping a connection $p = 0.5$), we would get N different predictions. We compute the *mean* and the *variance* of the N predictions for each sleep stage k

$$\mu_{\mathrm{i,k}} = \frac{\sum_{n=1}^{N} \hat{p}_{\mathrm{n,i,k}}}{N} \tag{10}$$

TABLE II
NUMBER AND PERCENTAGE OF 30-SECOND EPOCHS PER SLEEP STAGE OF THE SLEEP-EDF DATASETS WITH DIFFERENT TRIMMING

Dataset	W	N1	N2	N3	R	Total
v1-2013 ±30mins	8285 (19.6%)	2804 (6.6%)	17799 (42.1%)	5703 (13.5%)	7717 (18.2%)	42308
v1-2013	5907 (15.2%)	2687 (6.9%)	17255 (44.3%)	5465 (14.0%)	7647 (19.6%)	38961
v2-2018 ±30mins	65951 (33.7%)	21522 (11.0%)	69132 (35.4%)	13039 (6.7%)	25835 (13.2%)	195479
v2-2018	43055 (26.3%)	19168 (11.7%)	64408 (39.3%)	12042 (7.3%)	25275 (15.4%)	163948

$$\sigma^2_{\mathrm{i,k}} = \frac{\sum_{n=1}^{N} (\hat{p}_{\mathrm{n,i,k}} - \mu_{\mathrm{i,k}})^2}{N} \tag{11}$$

where $\hat{p}_{\mathrm{n,i,k}}$ is the output probability for the sleep stage k of the n-th prediction for the input $\mathbf{x}_{\mathrm{i}}$. The final prediction $\hat{y}_{\mathrm{i}}$ of the model will be given by $max(\boldsymbol{\mu}_{\mathrm{i}})$.

The uncertain predictions will be then estimated by analysing both their computed *mean* and *variance*. The selection procedure of the uncertain sleep stages is explained in detail in subsection VI-D. The selected uncertain predictions could be then presented to the physician for a secondary review.

V. DATA

Sleep-EDF (SC). The Sleep-EDF Sleep Cassette is a subset of the open source Sleep-EDF dataset [37]. The PSG data belong to 78 subjects (37 males and 41 females) aged from 25 to 101 years. Except for the first nights of subjects 36 and 52, and for the second night of subject 13, for all the subjects are available two whole nights, resulting in 153 PSG recordings. Each recording includes two scalp EEG channels (Fpz-Cz and Pz-Cz), one EOG (horizontal) channel, one submental chin EMG channel and one oro-nasal respiration channel. The recordings are manually scored by sleep experts on 30-second epochs according to Rechtschaffen and Kales scoring rules [38], resulting in the eight classes Wake, N1, N2, N3, N4, REM, MOVEMENT and UNKNOWN. In order to use the AASM standard [3], we have merged the N3 and N4 stages into a single stage N3, and we have excluded the MOVEMENT and UNKNOWN classes. In many recordings there were long wake periods before the patients went to sleep and after they woke up. We have done experiments with the two common ways these periods are trimmed in literature: 1) only *in-bed* parts are employed [4], i.e. from *light-off* time to *light-on* time; 2) 30 minutes of data before and after *in-bed* parts are taken into account in the experiments [18]. In our study we have considered the EEG Fpz-Cz channel, with a sampling rate of 100 Hz and without any pre-processing.

In order to facilitate the comparison with many existing deep learning based scoring algorithms, in this work we use the last expanded version published in 2018, and also the previous upload of the Sleep-EDF database published in 2013. In the older upload there were only 39 PSG recordings from 20 subjects. In Table II we report a summary of the total number and percentage of the epochs per sleep stage.

TABLE III
SUMMARY OF THE SLEEP-EDF DATASET AND THE DATA SPLIT

Dataset	Size	Experimental Setup	Held-out Validation Set	Held-out Test Set
v1-2013	20	20-fold CV	4 subjects	1 subject
v2-2018	78	20-fold CV	7 subjects	7 subjects

TABLE IV
OVERALL PERFORMANCE AND CALIBRATION MEASURE OF THE MODELS OBTAINED FROM 20-FOLD CROSS-VALIDATION WITH AND WITHOUT *MC* ON SLEEP-EDF v1-2013 ±30mins DATASET. BEST SHOWN IN BOLD

		Overall Metrics				Calibration	
	Models	Acc.	MF1	k	F1	ECE	*conf*
w/o MC	*base*	82.3	76.6	0.76	82.5	0.111	93.4
	base+LS_u	82.8	77.2	0.77	83.0	**0.023**	80.5
	base+LS_s	82.7	76.4	0.76	82.7	0.071	89.7
w/ MC	*base*	83.0	77.1	0.77	83.0	0.060	89.0
	***base*+LS_u**	**84.0**	**78.0**	**0.78**	**83.9**	0.055	78.5
	base+LS_s	83.4	77.0	0.77	83.3	**0.031**	86.5

VI. RESULTS

A. Experiment Design

Our validation procedure is in line with the state-of-the-art methods considered in Table VIII in subsection VI-E. In fact, we evaluate our model using the k-fold cross-validation scheme. We set k equal to 20 for v1-2013 and 10 for v2-2018 Sleep-EDF datasets. In Table III we summarize the data split for each dataset. We decide to further standardize the experiments by considering in each fold the same subject IDs used in [26]. We believe that in such small datasets, the subjects involved in the training/validation/test set may have an impact on the final results.

The following experiments are conducted:

- ***base***. The model is trained not considering model calibration, and without label smoothing.
- ***base*+LS_u**. The model is trained taking into account the confidence calibration, using *label smoothing* with *uniform distribution* - i.e. the hard targets are weighted with the *uniform distribution*.
- ***base*+LS_s**. The model is trained taking into account the confidence calibration, using *label smoothing* with our statistical analysis done on the sequences of sleep stages - i.e. the hard targets are weighted with the *conditional probability distribution*.

These three models, differently trained, have been evaluated with and without using the *MC dropout* sampling technique. In Table IV subsection VI-C we present the results obtained for the three models, and the impact of *MC dropout* at inference time.

The models have been implemented in TensorFlow 1.14, and trained on a single workstation running Ubuntu 18.04.2 with a Intel Core i7-8700K CPU, an NVIDIA GTX 1080 GPU with 8 GB memory and 32 GB RAM memory.

B. Metrics

Performance. The per-class F1-score, the overall accuracy (*Acc.*), the macro-averaging F1-score (*MF1*) and the Cohen's kappa (k) have been computed from the predicted sleep stages from all the folds to evaluate the performance of our model [39], [40]. In our experiments the weighted-averaging F1-score has been also reported, taking into account also the label imbalance problem. It computes the average of the metric weighted by the number of true instances for each label. The F1-score computed in this way is not a realistic weighted average of the precision and recall, but it takes into account the high imbalance between the sleep stages.

Calibration. We evaluated the calibration of our model using the expected calibration error (ECE) proposed in [41]. It approximates the difference in expectation between accuracy acc and confidence $conf$, where with confidence it refers to the softmax output probabilities.

More in detail, we first divide the predictions into M equally spaced bins (size $1/M$), then for each bin we compute the accuracy $acc(B_m)$ and we define the average predicted probability value $conf(B_m)$:

$$acc(B_m) = \frac{1}{|B_m|} \cdot \sum_{i \in B_m}^{K} \mathbf{1}(\hat{y}_i = y_i) \tag{12}$$

$$conf(B_m) = \frac{1}{|B_m|} \cdot \sum_{i \in B_m}^{K} \hat{p}_i \tag{13}$$

where y_i and $\hat{y}_i$ are the true and predicted labels for the sample i, B_m is the group of samples whose predicted probability values falls into the interval $I_m = (\frac{m-1}{M}, \frac{m}{M}]$, and $\hat{p}_i$ is the predicted probability value for sample i.

Then we finally compute the weighted average of the acc and $conf$ difference of the M bins,

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} \cdot |acc(B_m) - conf(B_m)| \tag{14}$$

where n is the number of samples in each bin.

Clearly, perfectly calibrated models have $acc(B_m) = conf(B_m)$ for all $m \in \{1, .., M\}$, resulting in $ECE = 0$.

C. Analysis of Experiments

In table IV we report the overall performance and the calibration measure of three different models, with and without *Monte Carlo dropout* at inference time, to which we refer *w/o MC* and *w/ MC* respectively. In the following, we analyse only the results obtained on the Sleep-EDF v1-2013 ±30mins dataset, since the findings are still valid for its expanded v2-2018 ±30mins version.

In our tests *w/o MC*, we show the efficiency of label smoothing in calibrating the model. The $conf$ value refers to the average of all the predicted probability values. In both LS_u and LS_s models, the $conf$ probability better reflects the ground truth correctness likelihood - i.e. accuracy value. Indeed, it results in a better ECE value 0.023 and 0.071, compared to the higher 0.111 for the *base* model. The overall performance are preserved or even improved.

By using *MC* at test time, we show the efficiency of label smoothing and *MC* techniques in both calibrating and enhancing the performance of the model. It is quite interesting the impact of *MC dropout*: an increase in overall metrics and

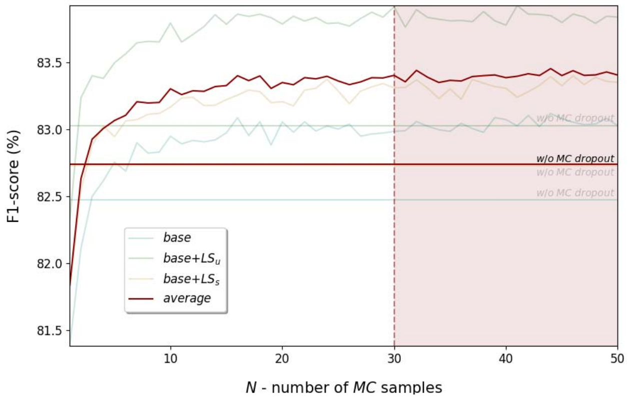

Fig. 3. F1-score against the number of *Monte Carlo* samples *N* of the three models (*base*, *base*+*LS*$_u$ and *base*+*LS*$_s$) evaluated on Sleep-EDF v1-2013 dataset. *Monte Carlo* sampling converges after 30 samples without further significant improvement on the average of the three models.

TABLE V
Confusion Matrix Obtained From 20-Fold Cross-Validation on Sleep-EDF v1-2013 ±30mins Dataset

	Predicted					Per-class Metrics		
	W	N1	N2	N3	R	Pr.	Rec.	F1
W	**90.4**	6.0	1.4	0.3	1.9	84.0	90.4	87.1
N1	18.6	**42.6**	19.2	0.8	18.8	46.5	42.6	44.4
N2	3.0	2.4	**86.0**	4.5	4.1	89.9	86.0	87.9
N3	1.3	0.2	7.9	**90.6**	0	85.9	90.6	88.2
R	3.5	5.7	7.8	0.1	**82.9**	82.0	82.8	82.4

a decrease in the average predicted probability values. This justifies a better calibrated model by using our *conditional probability distribution* smoothing technique *LS*$_s$ - i.e. ECE value equal to 0.031.

In Fig. 3 we report the F1-score against the number of *Monte Carlo* samples *N*, evaluated over all our experiments. Interesting how *Monte Carlo* sampling outperforms the experiments done without applying *MC* after approximately three samples, on the average of the three models. On average we get a plateau after 30 samples, so we decided to set *N* equal to 30.

From here on, all the results will refer to the best of our models *base*+*LS*$_u$, by using *MC* sampling at test time.

In Tables V and VI we report the confusion matrix and the per-class performance of the best of our models evaluated on Sleep-EDF v1-2013 ±30mins and v2-2018 ±30mins respectively. The *i-th* row and the *j-th* column indicates the percentage number of 90-s EEG instances with the true label being *i-th* class and the predicted label being *j-th* class. In bold we highlight the percentage number of instances well classified. As expected [42], the lowest performance has been obtained for the N1 sleep stage, i.e. F1-score 44.4% and 46.0%; most of the N1 have been wrongly classified in awake, N2 and REM. The F1-score for all the other sleep stages were in range between 82.4% and 88.2% on v1-2013 ±30mins, and between 76.4% and 91.5% on v2-2018 ±30mins.

D. Uncertainty Estimate

MC dropout enables the estimate of the uncertain predictions. In order to select the uncertain instances, at first, we used the *variance* (σ^2 of the predicted probability values

TABLE VI
Confusion Matrix Obtained From 10-Fold Cross-Validation on Sleep-EDF v2-2018 ±30mins Dataset

	Predicted					Per-class Metrics		
	W	N1	N2	N3	R	Pr.	Rec.	F1
W	**90.0**	7.5	0.6	0.2	1.7	93.0	90.0	91.5
N1	14.2	**48.1**	24.5	1.0	12.2	44.1	48.1	46.0
N2	0.7	8.5	**80.3**	5.3	5.2	85.6	80.3	82.9
N3	0.2	0.3	12.8	**86.5**	0.2	73.0	86.5	79.2
R	3.4	8.8	7.6	0.8	**79.4**	73.7	79.4	76.4

TABLE VII
Per-Class σ^2 and μ of the Predicted Probability Values Computed on Sleep-EDF v1-2013 ±30mins and v2-2018 ±30mins Datasets

Dataset	Total	W	N1	N2	N3	R
v1-2013	σ^2	0.008	0.024	0.009	0.006	0.014
±30mins	μ	81.3	60.2	80.4	81.7	75.3
v2-2018	σ^2	0.005	0.015	0.009	0.006	0.013
±30mins	μ	82.9	60.4	74.7	79.5	70.4

obtained from the N sampling). The selection procedure (also referred to as *query* procedure) simply rely on the setting of a threshold value $q\%$, that corresponds to the percentage number of epochs - for each PSG recording - to select and to send potentially to the physician for a secondary review. The epochs with the highest values of *variance* will be the $q\%$ selected. We also tried to use the *mean* (μ of the predicted probability values obtained from the N sampling) to select the uncertain instances. In this case the epochs with the lowest *mean* values will be the $q\%$ selected.

The selected epochs, in both cases, correspond to the predictions where the averaging ensemble of the models outputs the higher uncertainty. In Fig. 4 we report the F1-score computed over the remaining epochs against the percentage number of selected instances. We have fixed the $q\%$ threshold value to 5%, because it was considered to be a reasonable number of epochs (54 on average for each PSG recording) to select and to eventually present to the physician for a secondary review. The results show that by using μ in the selection procedure we obtain higher performance. In Fig. 5 we also report, for each $q\%$ number of selected instances, the percentage of misclassified and correctly classified epochs among the selected ones. As illustrated, by using μ, the percentage number of misclassified epochs are greater than the correctly classified up to the selection threshold $q\%$ equal to 10%. Whilst, by using σ^2, the percentage number of selected epochs $q\%$ radically decreases to 2%.

In Table VII we also report the average of the per-class σ^2 and μ predicted probability values, to have an overall estimate of the model uncertainty, evaluated on both Sleep-EDF v1-2013 ±30mins and v2-2018 ±30mins datasets. As expected, the results show that the model has more difficulty in classifying N1 and REM epochs, while provides greater confidence in classifying W, N2 and N3 sleep stages (lower variance and higher predicted probability values).

E. Comparison With State-of-the-Art

In Table VIII we compare our best model with the other state-of-the-art methods evaluated on the two versions of the

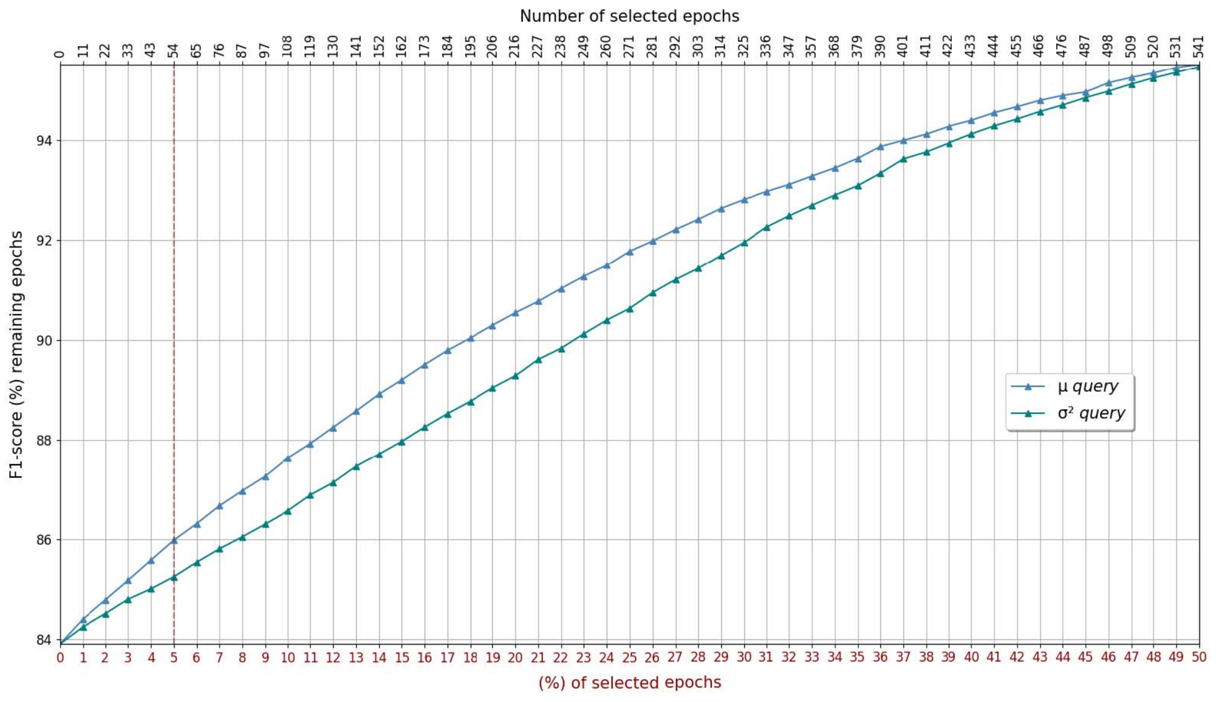

Fig. 4. F1-score computed over the remaining epochs after the *query* procedure against the percentage number of epochs to select. In light green and in light blue the F1-score performance in case the selection procedure has been done using the variance (σ^2 *query*) and the mean (μ *query*) respectively. The performance refers to the best of our model evaluated on Sleep-EDF v1-2013 ±30mins dataset.

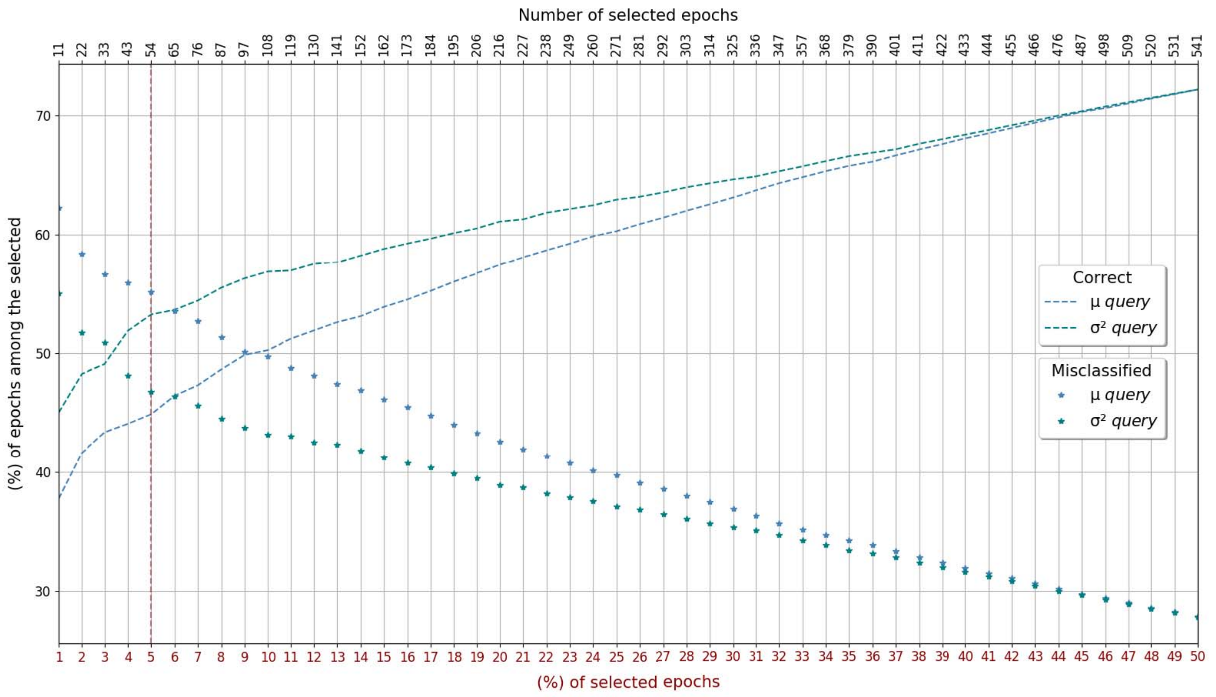

Fig. 5. Percentage of misclassified and correctly classified epochs among the q% selected. In light green and in light blue the percentage values in case the selection procedure has been done using the variance (σ^2 *query*) and the mean (μ *query*) respectively. The performance refers to the best of our models evaluated on Sleep-EDF v1-2013 ±30mins dataset.

Sleep-EDF database. We report the results for each experimental scenario: 1) only *in-bed* recordings; 2) additional 30 minutes recordings before and after *in-bed*. We have considered only the methods using deep learning based architectures, raw single channel Fpz-Cz, same evaluation procedure (i.e. k-fold cross-validation) and using independent training and test sets. We decided to further standardize our experiments by considering in each fold the same subject IDs used in [26].

TABLE VIII

Comparison Between Our Method and the Other Deep Learning-Based Automatic Sleep Scoring Systems Using Raw Single Channel Fpz-Cz, Evaluated on Sleep-EDF Datasets With Overall Accuracy (Acc.), Macro F1-Score (MF1), Cohen's Kappa (k) and per-Class F1-Score. The Best Performance Metrics for Each Dataset Are Indicated in Bold

Datasets	Methods	Training	Sequences	Overall Metrics			Per-Class F1-Score				
		Param.	of epochs	Acc.	MF1	k	W	N1	N2	N3	R
v1-2013 ±30mins	FCNN+RNN [26]	∼ 5.6M	20	81.8	75.6	0.75	89.4	44.1	84.0	84.0	76.3
	DeepSleepNet [18]	∼ 24.7M	25	81.4	75.6	0.75	83.9	43.5	85.4	84.1	81.1
	DeepSleepNet [18] †	∼ 24.7M	25	82.0	76.9	0.76	84.7	46.6	85.9	84.8	82.4
	IITNet [22] †	-	10	84.0	77.7	0.78	87.9	44.7	88.0	85.7	82.1
	Our method	∼ 0.6M	**3**	84.0	78.0	0.78	87.1	44.4	87.9	88.2	82.4
	SleepEEGNet [23] †	∼ 2.6M	10	84.3	79.7	0.79	89.2	**52.2**	86.8	85.1	**85.0**
	SeqSleepNet+ [24]	∼ **0.2M**	20	85.2	78.4	0.80	90.5	45.4	**88.1**	86.4	81.8
	Naive Fusion [26]	∼ 5.8M	20	85.0	78.8	0.79	91.7	48.8	87.2	82.9	83.6
	TinySleepNet [25] †	∼ 1.3M	15	85.4	80.5	0.80	90.1	51.4	88.5	**88.3**	84.3
	XSleepNet2 [26]	∼ 5.8M	20	**86.3**	**80.6**	**0.81**	**92.2**	51.8	88.0	86.8	83.9
v1-2013	Naive Fusion [26]	∼ 5.8M	20	80.2	74.9	0.72	77.3	47.4	85.8	84.8	79.3
	DeepSleepNet [18]	∼ 24.7M	25	82.5	76.8	0.76	80.1	47.3	87.0	85.7	83.8
	DeepSleepNet [18] †	∼ 24.7M	25	82.6	77.1	0.76	**82.9**	46.8	86.5	84.1	85.2
	FCNN+RNN [26]	∼ 5.6M	20	81.3	76.0	0.74	76.4	50.0	86.8	85.3	81.3
	SeqSleepNet+ [24]	∼ **0.2M**	20	82.2	74.1	0.75	78.5	37.1	87.6	86.2	81.2
	Our method	∼ 0.6M	**3**	82.6	76.3	0.76	81.6	42.4	87.4	**87.9**	82.1
	XSleepNet2 [26]	∼ 5.8M	20	**83.9**	**78.7**	**0.77**	81.6	**52.9**	**88.1**	85.3	**85.4**
v2-2018 ±30mins	DeepSleepNet [18]	∼ 24.7M	25	76.9	70.7	0.69	90.8	44.8	78.5	67.9	71.3
	DeepSleepNet [18] †	∼ 24.7M	25	77.1	71.2	0.69	90.4	46.0	79.1	68.6	71.8
	SleepEEGNet [23] †	∼ 2.6M	10	80.0	73.6	0.73	91.7	44.1	82.5	73.5	76.1
	Our method	∼ 0.6M	**3**	80.3	75.2	0.73	91.5	46.0	82.9	79.2	76.4
	Naive Fusion [26]	∼ 5.8M	20	82.3	76.2	0.75	93.2	49.6	**86.2**	79.4	**82.5**
	SeqSleepNet+ [24]	∼ **0.2M**	20	82.6	76.4	0.76	92.2	47.8	84.9	77.2	79.9
	FCNN+RNN [26]	∼ 5.6M	20	82.8	76.6	0.76	92.5	47.3	85.0	79.2	78.9
	TinySleepNet [25] †	∼ 1.3M	15	83.1	**78.1**	0.77	92.8	**51.0**	85.3	**81.1**	80.3
	XSleepNet2 [26]	∼ 5.8M	20	**84.0**	77.9	**0.78**	**93.3**	49.9	86.0	78.7	81.8
v2-2018	DeepSleepNet [18]	∼ 24.7M	25	76.0	72.2	0.68	88.1	45.8	79.7	74.3	72.9
	DeepSleepNet [18] †	∼ 24.7M	25	76.6	73.0	0.69	88.3	46.1	79.9	76.2	74.4
	SeqSleepNet+ [24]	∼ **0.2M**	20	79.0	74.6	0.71	83.2	46.8	85.5	76.3	81.0
	Our method	∼ 0.6M	**3**	79.0	75.1	0.72	**89.3**	46.9	83.3	78.9	77.1
	Naive Fusion [26]	∼ 5.8M	20	79.1	75.1	0.71	83.9	47.8	85.4	78.4	79.8
	FCNN+RNN [26]	∼ 5.6M	20	79.3	75.1	0.71	84.2	49.1	85.2	76.8	80.4
	XSleepNet2 [26]	∼ 5.8M	20	**80.3**	**76.4**	**0.73**	85.2	**49.4**	**86.0**	**79.8**	**81.7**

All the results indicated by † are not directly comparable, since they use a different set of subject IDs in their training/ evaluation/ testing procedure. The sleep scoring algorithms are compared across the overall metrics (Acc., MF1, Cohen's Kappa and F1-score) and the per-class F1-score. The proposed DeepSleepNet-Lite achieves slightly lower performance, if not on par, compared to the state-of-the-art models on all the Sleep-EDF datasets. The results confirm what we had already partially observed in [27] on the Sleep-EDF v1-2013: the first *epoch processing block* from DeepSleepNet, trained with a small temporal context in input, still succeed in solving the classification task on the small-sized database. Indeed, on both v1-2013 and v2-2018 *in-bed* recordings, our model achieves an overall accuracy only below 1.3% compared to the recent state-of-the-art XSleepNet2 [26]. We are not surprised to see our lighter architecture to overperform DeepSleepNet: one of the reasons could be that in [18] they have not implemented any early stopping procedure, and they save their model only at the latest iteration step, thus not mitigating the overfitting phenomenon. The number of training parameters of our lighter model are significantly reduced, ∼0.6M compared to the others TinySleepNet [25] ∼1.3M, SleepEEGNet [23] ∼2.6M, FCNN+RNN [26] ∼5.6M, Naive Fusion and XSleepNet2 ∼5.8M [26] and DeepSleepNet [18] ∼24.7M. Nevertheless, SeqSleepNet [24] is still the network with the lowest number of parameters ∼0.2M. We did not report the number of training parameters for IITNet [22] since it was not available in literature.

F. Comparison Among Our Methods

In Table IX we report the results of our best model evaluated on the two versions of the Sleep-EDF database - in both experimental scenarios. The outcomes refer to the performance of the model evaluated before the selection procedure and after the selection procedure, by using σ^2 and μ *query* values. We report the results obtained after the selection procedure on both the kept and rejected set of epochs. As a consequence of what we have observed in Fig. 5, on both Sleep-EDF v1-2013 and v2-2018, the model shows an increase in performance over the kept epochs, and a significant decrease on the rejected epochs (below 50% by using μ *query*). These results highlight the efficiency of the *query* procedure to select a larger number of misclassified epochs among the selected one. The best performance for each dataset are indicated in bold. We obtain an overall accuracy equal to 86.1% on v1-2013 ±30mins (84.5% on *in-bed* only) and equal to 82.3% on v2-2018 ±30mins 80.9% on *in-bed* only).

TABLE IX

COMPARISON AMONG OUR METHODS USING RAW SINGLE CHANNEL FPZ-CZ, EVALUATED ON SLEEP-EDF DATASETS WITH OVERALL ACCURACY (ACC.), MACRO F1-SCORE (MF1), COHEN'S KAPPA (κ), WEIGHTED-AVERAGING F1-SCORE (F1) AND PER-CLASS F1-SCORE. THE BEST PERFORMANCE METRICS FOR EACH DATASET ARE INDICATED IN BOLD

Datasets	Selection Procedure	Evaluated Epochs	Overall Metrics Acc.	MF1	k	F1	Per-Class F1-Score W	N1	N2	N3	R
v1-2013 ±30mins	-	all	84.0	78.0	0.78	83.9	87.1	44.4	87.9	88.2	82.4
	σ^2 *query*	kept	85.7	77.9	0.80	85.2	88.6	39.6	88.9	88.5	84.2
		rejected	53.0	47.5	0.36	52.4	38.7	54.9	53.3	32.3	58.1
	μ *query*	kept	**86.1**	**79.6**	**0.81**	**86.0**	**89.1**	**45.7**	**89.4**	**89.0**	**84.8**
		rejected	44.7	43.4	0.28	44.8	42.8	39.3	49.2	37.9	47.8
v1-2013	-	all	82.6	76.3	0.76	82.4	81.6	42.4	87.4	87.9	82.1
	σ^2 *query*	kept	84.3	76.4	0.78	83.8	83.5	37.9	88.4	88.3	83.6
		rejected	50.0	45.2	0.32	49.4	39.6	51.9	47.2	30.6	56.8
	μ *query*	kept	**84.5**	**77.8**	**0.78**	**84.4**	**83.6**	**43.5**	**88.8**	**88.7**	**84.4**
		rejected	45.6	45.7	0.29	45.8	45.9	37.6	52.2	49.3	43.5
v2-2018 ±30mins	-	all	80.3	75.2	0.73	80.6	91.5	46.0	82.9	79.2	76.4
	σ^2 *query*	kept	81.7	75.9	0.75	81.8	92.3	45.6	84.0	79.9	77.5
		rejected	55.2	51.0	0.40	54.4	49.9	49.0	48.1	39.5	68.6
	μ *query*	kept	**82.3**	**76.7**	**0.76**	**82.5**	**92.6**	**47.1**	**84.4**	**80.1**	**79.4**
		rejected	42.8	41.8	0.24	43.0	44.3	39.4	45.8	35.6	43.8
v2-2018	-	all	79.0	75.1	0.72	79.3	89.3	46.9	83.3	78.9	77.1
	σ^2 *query*	kept	80.2	75.8	0.73	80.4	90.0	46.9	84.3	79.8	77.8
		rejected	57.1	52.1	0.42	56.4	48.9	47.7	50.7	41.4	71.7
	μ *query*	kept	**80.9**	**76.7**	**0.74**	**81.2**	**90.7**	**48.0**	**84.7**	**79.8**	**79.7**
		rejected	42.4	41.1	0.23	42.6	41.3	39.7	44.9	34.6	44.9

VII. DISCUSSION

Our simplified deep learning approach to sleep scoring achieves performance slightly lower, if not on par, compared to the existing state-of-the-art methodologies evaluated on the Sleep-EDF database. Besides being trained on a small number of parameters, our method does not require any extra resources to buffer the sequences in input, since it processes sequences of only 90-seconds EEG. Therefore, we may assume that an automatic sleep scoring system does not necessarily have to encode such long temporal structures, rather intrinsic patterns of short-term PSG recordings may be sufficient.

However, as a result of further experiments carried out on larger and more heterogeneous databases (e.g. Physio2018 [43], [44] and SHHS [45], [46]), we can state that these observations are mainly valid on small-sized dataset (i.e. low heterogeneity between subjects).

The major advantage of the proposed approach is that it also provides an estimate of the model uncertainty by exploiting existing layers of the architecture. Unlike the existing confidence estimation algorithms for sleep scoring [14], [21], the *Monte Carlo dropout* is easy to implement and it does not require any additional computation over the baseline architecture. Moreover, it produces interpretable outputs, i.e. *mean* and *variance* of the predicted probability values. A clear disadvantage for this approach - as for other ensemble learning based algorithms - is that it needs to be executed N times, obviously increasing the evaluation time by N. However, in a real-time application, it may still be a valid solution because the evaluation of a single sequence takes only a few milliseconds.

The results obtained in subsection VI-C, in case our model is trained by smoothing the labels through the *conditional probability distribution*, are still to be further investigated. The impact of this prior knowledge, inserted during the training of our architecture, is not so obvious. It seems to improve the calibration process of the model while maintaining its overall good performance. Even if with this technique we succeed to better calibrate our network, we do not equally succeed in obtaining higher performance using it in combination with *Monte Carlo dropout*. Therefore, unlike what we expected, it is not always the case that a better calibrated architecture leads to higher performance, or even, to a better estimate of the model uncertainty.

VIII. CONCLUSION AND FUTURE WORKS

We propose DeepSleepNet-Lite a simplified and lightweight automatic sleep scoring architecture, providing the predicted sleep stages along with an estimate of their uncertainty. The scoring system is based on raw single channel EEG, and it processes 90-seconds time sequences. Although the proposed simple feed forward architecture has proven to be as efficient as RNNs based architectures, we cannot conclude that by using only this first representation learning block we will reach equally good results on larger databases. The *Monte Carlo dropout* technique allows us to enhance the performance of the architecture and to identify a relevant number of misclassified epochs among the ones selected during the query procedure. DeepSleepNet-Lite has a low capacity, i.e. low number of training parameters, hence is less prone to overfitting on a small dataset. Therefore the need to further investigate its robustness on larger database. It would be interesting to simulate the query procedure on the recent state-of-the-art architectures, e.g. XSleepNet2, to assess its benefit on them. Our lightweight sleep scoring approach paves the way to real-time applications and to home-monitoring scenarios.

REFERENCES

[1] M. M. Ohayon, "Epidemiological overview of sleep disorders in the general population," *Sleep Med. Res.*, vol. 2, no. 1, pp. 1–9, Apr. 2011.

[2] T. Penzel and R. Conradt, "Computer based sleep recording and analysis," *Sleep Med. Rev.*, vol. 4, no. 2, pp. 131–148, 2000.

[3] C. Iber, *The AASM Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and Technical Specifications*, vol. 1. Westchester, IL, USA: American Academy of Sleep Medicine, 2007.

[4] S. A. Imtiaz and E. Rodriguez-Villegas, "An open-source toolbox for standardized use of PhysioNet sleep EDF expanded database," in *Proc. 37th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2015, pp. 6014–6017.

[5] C. Berthomier *et al.*, "Exploring scoring methods for research studies: Accuracy and variability of visual and automated sleep scoring," *J. Sleep Res.*, vol. 29, no. 5, Oct. 2020, Art. no. e12994.

[6] L. Fiorillo *et al.*, "Automated sleep scoring: A review of the latest approaches," *Sleep Med. Rev.*, vol. 48, Dec. 2019, Art. no. 101204.

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

[8] H. Dong, A. Supratak, W. Pan, C. Wu, P. M. Matthews, and Y. Guo, "Mixed neural network approach for temporal sleep stage classification," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 26, no. 2, pp. 324–333, Feb. 2018.

[9] O. Tsinalis, P. M. Matthews, Y. Guo, and S. Zafeiriou, "Automatic sleep stage scoring with single-channel EEG using convolutional neural networks," 2016, *arXiv:1610.01683*. [Online]. Available: http://arxiv.org/abs/1610.01683

[10] A. Vilamala, K. H. Madsen, and L. K. Hansen, "Deep convolutional neural networks for interpretable analysis of EEG sleep stage scoring," in *Proc. IEEE 27th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Sep. 2017, pp. 1–6.

[11] S. Chambon, M. N. Galtier, P. J. Arnal, G. Wainrib, and A. Gramfort, "A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 26, no. 4, pp. 758–769, Apr. 2018.

[12] Z. Cui, "Automatic sleep stage classification based on convolutional neural network and fine-grained segments," *Complexity*, vol. 2018, Dec. 2018, Art. no. 9248410.

[13] A. N. Olesen, P. Jennum, P. Peppard, E. Mignot, and H. B. D. Sorensen, "Deep residual networks for automatic sleep stage classification of raw polysomnographic waveforms," in *Proc. 40th Annu. Int. Conf. Eng. Med. Biol. Soc. (EMBC)*, Jul. 2018, pp. 1–4.

[14] A. Patanaik, J. L. Ong, J. J. Gooley, S. Ancoli-Israel, and M. W. L. Chee, "An end-to-end framework for real-time automatic sleep stage classification," *Sleep*, vol. 41, no. 5, May 2018, Art. no. zsy041.

[15] A. Sors, S. Bonnet, S. Mirek, L. Vercueil, and J.-F. Payen, "A convolutional neural network for sleep stage scoring from raw single-channel EEG," *Biomed. Signal Process. Control*, vol. 42, pp. 107–114, Apr. 2018.

[16] O. Yildirim, U. Baloglu, and U. Acharya, "A deep learning model for automated sleep stages classification using PSG signals," *Int. J. Environ. Res. Public Health*, vol. 16, no. 4, p. 599, Feb. 2019.

[17] N. Michielli, U. R. Acharya, and F. Molinari, "Cascaded LSTM recurrent neural network for automated sleep stage classification using single-channel EEG signals," *Comput. Biol. Med.*, vol. 106, pp. 71–81, Mar. 2019.

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

[19] S. Biswal, H. Sun, B. Goparaju, M. B. Westover, J. Sun, and M. T. Bianchi, "Expert-level sleep scoring with deep neural networks," *J. Amer. Med. Inform. Assoc.*, vol. 25, no. 2, pp. 1643–1650, 2018.

[20] A. Malafeev *et al.*, "Automatic human sleep stage scoring using deep neural networks," *Frontiers Neurosci.*, vol. 12, p. 781, Nov. 2018.

[21] J. B. Stephansen, "Neural network analysis of sleep stages enables efficient diagnosis of narcolepsy," *Nature Commun.*, vol. 9, no. 1, pp. 1–15, 2018.

[22] H. Seo, S. Back, S. Lee, D. Park, T. Kim, and K. Lee, "Intra- and inter-epoch temporal context network (IITNet) using sub-epoch features for automatic sleep scoring on raw single-channel EEG," 2019, *arXiv:1902.06562*. [Online]. Available: http://arxiv.org/abs/1902.06562

[23] S. Mousavi, F. Afghah, and U. Acharya, "SleepEEGNet: Automated sleep stage scoring with sequence to sequence deep learning approach," *PLoS ONE*, vol. 14, no. 5, 2019, Art. no. e0216456.

[24] H. Phan *et al.*, "Towards more accurate automatic sleep staging via deep transfer learning," *IEEE Trans. Biomed. Eng.*, vol. 68, no. 6, pp. 1787–1798, Jun. 2021.

[25] A. Supratak and Y. Guo, "TinySleepNet: An efficient deep learning model for sleep stage scoring based on raw single-channel EEG," in *Proc. 42nd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2020, pp. 641–644.

[26] H. Phan, O. Y. Chen, M. C. Tran, P. Koch, A. Mertins, and M. De Vos, "XSleepNet: Multi-view sequential model for automatic sleep staging," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Mar. 31, 2021, doi: 10.1109/TPAMI.2021.3070057.

[27] L. Fiorillo, M. Wand, I. Marino, P. Favaro, and F. D. Faraci, "Temporal dependency in automatic sleep scoring via deep learning based architectures: An empirical study," in *Proc. 42nd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2020, pp. 3509–3512.

[28] M. X. Cohen, *Analyzing Neural Time Series Data: Theory and Practice*. Cambridge, MA, USA: MIT Press, 2014.

[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[32] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks Trade*. Berlin, Germany: Springer, 1998, pp. 55–69.

[33] I. Goodfellow, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[35] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?" in *Proc. Adv. Neural Inf. Process. Systems.*, 2019, pp. 4694–4703.

[36] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.

[37] B. Kemp, A. H. Zwinderman, B. Tuk, H. A. C. Kamphuisen, and J. J. L. Oberye, "Analysis of a sleep-dependent neuronal feedback loop: The slow-wave microcontinuity of the EEG," *IEEE Trans. Biomed. Eng.*, vol. 47, no. 9, pp. 1185–1194, Sep. 2000.

[38] E. A. Wolpert, "A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects," *Arch. Gen. Psychiatry*, vol. 20, no. 2, p. 246, Feb. 1969.

[39] J. Cohen, "A coefficient of agreement for nominal scales," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, 1960.

[40] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, 2009.

[41] M. Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using Bayesian binning," in *Proc. Conf. Artif. Intell.*, 2015, p. 2901.

[42] R. S. Rosenberg and S. Van Hout, "The American academy of sleep medicine inter-scorer reliability program: Sleep stage scoring," *J. Clin. Sleep Med.*, vol. 9, no. 1, pp. 81–87, Jan. 2013.

[43] A. L. Goldberger *et al.*, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, Jun. 2000.

[44] M. Ghassemi *et al.*, "You snooze, you win: The PhysioNet/Computing in cardiology challenge 2018," in *Proc. Comput. Cardiol. Conf. (CinC)*, Dec. 2018, pp. 1–4.

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

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