Published in final edited form as:
Conf Proc IEEE Int Conf Syst Man Cybern. 2023 October ; 2023: 2315–2320. doi:10.1109/SMC53992.2023.10394274.

MorpheusNet: Resource efficient sleep stage classifier for embedded on-line systems

Ali Kavoosi[1,✉], **Morgan P. Mitchell**[2], **Raveen Kariyawasam**[3], **John E. Fleming**[1], **Penny Lewis**[4], **Heidi Johansen-Berg**[2], **Hayriye Cagnan**[1], **Timothy Denison**[1,3]

[1]MRC Brain Network Dynamics Unit, University of Oxford, Oxford, UK

[2]Wellcome Centre for Integrative Neuroimaging, University of Oxford, Oxford, UK

[3]Department of Engineering Sciences, University of Oxford, Oxford, UK

[4]Brain Research Imaging Centre, Cardiff University, Cardiff, UK

Abstract

Sleep Stage Classification (SSC) is a labor-intensive task, requiring experts to examine hours of electrophysiological recordings for manual classification. This is a limiting factor when it comes to leveraging sleep stages for therapeutic purposes. With increasing affordability and expansion of wearable devices, automating SSC may enable deployment of sleep-based therapies at scale. Deep Learning has gained increasing attention as a potential method to automate this process. Previous research has shown accuracy comparable to manual expert scores. However, previous approaches require sizable amount of memory and computational resources. This constrains the ability to classify in real time and deploy models on the edge. To address this gap, we aim to provide a model capable of predicting sleep stages in real-time, without requiring access to external computational sources (e.g., mobile phone, cloud). The algorithm is power efficient to enable use on embedded battery powered systems. Our compact sleep stage classifier can be deployed on most off-the-shelf microcontrollers (MCU) with constrained hardware settings. This is due to the memory footprint of our approach requiring significantly fewer operations. The model was tested on three publicly available data bases and achieved performance comparable to the state of the art, whilst reducing model complexity by orders of magnitude (up to 280 times smaller compared to state of the art). We further optimized the model with quantization of parameters to 8 bits with only an average drop of 0.95% in accuracy. When implemented in firmware, the quantized model achieves a latency of 1.6 seconds on an Arm Cortex-M4 processor, allowing its use for on-line SSC-based therapies.

Index Terms

SSC; deep learning; resource efficient

✉ ali.kavoosi@linacre.ox.ac.uk .

This work was supported by the John Fell Fund of the University of Oxford, the UK Medical Research Council (MC UU 00003/3, MC UU 00003/6) and the Royal Academy of Engineering.

I Introduction

A Sleep architecture

Sleep consists of different stages. The identification of the different stages forms an important part of different clinical applications. Diagnosis of sleep conditions, studying the sleep pattern of patients, as well as applying therapies that require to be applied at certain times during sleep require the classification of the different stages. Specifically for adaptive therapies, a further requirement will be to classify the sleep stages in real time. This will be beneficial as there is evidence that demonstrates that adaptive therapies are effective in laboratory conditions [1].

Sleep stages are labeled manually by experts after examining electrophysiological recordings such as Electroen-cephalogram (EEG) from the scalp measuring the brain activity in a non-invasive manner, Electrocardiogram (ECG) measuring the heart's rhythm and Electrooculogram (EOG) capturing the eye movements. In a clinical setting, Polysomnography (PSG) is used to record various types of electrophysiological data which are then labeled by experts using various features defined in sleep manuals, such as the American Academy of Sleep Medicine (AASM) or the R & K standard. According to AASM, sleep can be separated into three stages: Non-Rapid Eye Movement (NREM) which consists of 3 different sub-stages in itself, Rapid Eye Movement (REM) and awake [2].

B Clinical Applications, Motivation for Requirements

Targeted memory reactivation (TMR) is an experimental approach which involves pairing a memory or an action (e.g. button presses), with sensory cues (e.g., sound) during a training session. Same sensory cues are then "replayed" during sleep to achieve targeted memory reactivation. Recent studies suggest that replaying previously paired auditory cues during sleep spindles, which are key features of NREM sleep, is optimal for memory consolidation [3] [4] [5], introducing a novel treatment approach for stroke rehabilitation to promote motor learning.

TMR has also been explored in the context of mood and psychiatric disorders, based on studies highlighting a reduction in emotional arousal due to embarrassing memories following TMR during REM sleep [6].

C Requirements 2: Scalable, at-home solution

TMR requires real-time classification of sleep stages and to-date has been carried out in sleep laboratories. This is a bottleneck for deploying such therapies at scale.

To address scalability and at-home deployment, there is a need to automate SSC and do so in real-time. There are different commercial products that perform SSC in an at-home environment [7]. However, these products are somewhat restrictive in terms of access to raw data and classification labels in real-time. As highlighted in Section B, predicted sleep stages should be accessible to implement TMR during specific sleep stages.

D Automated SSC in energy constrained environments

Deep Learning (DL) has been explored to automate SSC [8] [9]. Different DL architectures, such as Convolutional Neural Networks (CNNs) [10], Recurrent Neural Networks (RNNs) [11] and others have been experimented with in research. The DL-based models have achieved state of the art in terms of performance. These automated approaches have reached human level performance [12].

As the DL models require sizable amounts of data to learn from, recent introduction of publicly available datasets have enabled accurate DL models. However, continuing research in automating SSC has shown that previous models may have been more complex than required. Recent models with smaller size and lower complexity like tinySleepnet [13] have shown similar or even better performance compared to state of the art.

DL models can be used for a scalable solution to perform SSC in real-time. However, most of the models outlined to date, require sophisticated pieces of hardware [14] and transmission of data to cloud servers or mobile devices, which introduces cybersecurity threats. Cybersecurity threats have been addressed in Food and Drug Authority (FDA) approved medical devices [15]. Therefore, a compact system that can carry out inference locally would not only save cost and consume less energy, but also improve system security.

To meet these needs, we introduce MorphuesNet [16], a highly optimized and compact model for embedded online SSC to fill this gap. Table I summarizes the requirements for MorphuesNet which are addressed in this paper.

II Design Principles

We experiment with a architecture search approach. The findings are then used to design a compact network.

A Architecture search

We first experimented with light weight Neural Architecture Search (NAS) approaches. The search space was constrained to include two types of operations, depth-wise separable convolutions and normal convolutions for operations and max and average pooling for reduction. We deploy a Differentiable Architecture Search (DARTS) approach. However this is done by making the optimization of architecture parameters and model parameters as a single optimization problem, rather than the bi-level optimization approach used in the original paper [17]. The search is then further constrained to only have one node per layer. With this simplification, the optimization is learning a macro-architecture in a constrained search space as the architecture parameters are learned for the entire network and not for repetitive cells. To confine the model to choose between options that require less computation, the choice of kernel order and number of filters in the convolutional operations was also kept low. The maximum number of filters was fixed at 64 and the maximum kernel order was fixed at 32. The final architecture consists of four convolutional layers, and two reduction layers. This is outlined in Fig. 1. The choice of not learning sequences was made to further reduce the computational complexity. This means that the NAS will search only for feature extraction models.

With this approach, the edges, each of which represents an operation, are allocated an importance factor denoted as α. In this case, the optimization of the network learns both the importance of each edge and the parameters of the model denoted as θ.

Let X be in the input to a cell, consisting of operations o^1, o^2 and so on, where o^i is the i^{th} operation connecting the input to the output of that cell. The function of this cell would be. $Y = \{\alpha^1 o^1(X), \alpha^2 o^2(X), \alpha^3 o^3(X), \ldots\}$, where Y is the output of the cell. Simplifying this will yield:

$$Y = \alpha o(X) \tag{1}$$

The parameters are updated using gradient decent:

$$\alpha^* = \alpha - \eta \frac{\partial L(\theta, \alpha)}{\partial \alpha} \tag{2}$$

$$\theta^* = \theta - \eta \frac{\partial L(\theta, \alpha)}{\partial \theta} \tag{3}$$

where θ^* and α^* are the next set of model and architecture parameters to be used, $L(\theta, \alpha)$ is the loss calculated by the model and η is the learning rate. The choice of architecture is finalized by finding the operation with the highest α value between each two nodes.

Here, we observe that in the first convolutional cell, the architecture search allocates higher importance to features extracted with normal convolutions with bigger kernels. We hypothesize that this is to extract spatial features in the first layer, as there is no channel wise information in the raw input at the first convolutional operation. In this case, a normal convolutional filter can extract more useful features. As the goal is to upload the model on a restricted hardware platform, the possible operations were kept low in terms of computational power required. Within the domain of computationally efficient operations, the architecture search favors separable convolutions over normal convolutions. Another observation was that the model prefers max pooling in the earlier stages of the architecture and average pooling in the later stages. Fig. 1 shows α for two different convolutional operations in one of the cells used in the architecture search.

We used the findings from the Neural Architecture Search experiment to design a shallow residual network [18], based on depth-wise separable convolutions [19]. To address AASM guidelines regarding transitions between states, we added a sequence learning part to the CNN's learned features. To further reduce the number of operations and parameters, the sequence learning was applied to the output layer of the CNN, which denotes the probability of different states learned separately. This part includes a Long Short-Term Memory (LSTM) based layer of 32 nodes [20], a Dense layer with 32 nodes with Rectified Linear

Unit (ReLU) activation function [21], where dropout was applied between the layers with a 20 % dropout rate. Fig. 2 shows the final model architecture, referred to as MorpheusNet from hereon. The number of filters and kernel sizes were also derived from the architecture search results.

B Training

We divide the training in 2 processes, learning feature extraction and sequence learning. Feature learning is based on the CNN. The CNN was optimized using the Adam optimizer. The learning rate was set at 0.001 and β_1 was set to 0.9 and β_2 to 0.999. The data was shuffled and divided into mini-batches of 128 samples and the model was trained for 10 epochs. The training process uses categorical cross entropy loss to update parameters. First, the model was trained on the training set. After training, the model with the best performance on the validation set was selected. The metrics were calculated on the held-out test set. The input to the sequence learning model consists of concatenating the output of the CNN from 12 consecutive epochs. For training the sequence learner, The same training and validation set are used. This training process has the same Adam optimizer parameters as the CNN. However, the training set is divided into mini batches of 32 samples and learning rate is set at 0.0001. The metrics reported are from the test set.

C 8-bit Quantization: MorpheusNetQ

To further simplify the model, we also experimented quantizing the model parameters. Quantization was only applied on the CNN. The quantized CNN was fine tuned to the training data for 5 epochs. The sequence learner was then fine-tuned to the quantized CNN as well. This was done by using mini-batches of 32 samples and an Adam optimizer (learning rate was 0.001) for 5 epochs. This model will be referred to as MorphuesNetQ.

III Dataset and Data Preparation

We test the model on three publicly available datasets, Sleep-EDF [22], Dreem Open Dataset (DOD) [12] and Physionet 2018 challenge [23]. Sleep-EDF database has two subsets, Sleep Cassette (SC) and Sleep Telemetry (ST). SC was recorded from 78 participants. The EEG channel FPz-Cz was used for classification. The dataset was expanded in 2018 from it's original 20 subjects. However, for comparison purposes, we also use the 20 subjects subset as well. For fairness in comparison, we use the same subsets for k-fold cross validation used in [8]. In the k-fold cross validation experiment, k was set to 20 for Sleep-EDF 20 and 10 for Sleep-EDF 78. Sleep-EDF was annotated according to the R&K manual. R&K standard divides sleep stages into 6 stages (N4,N3,N2,N1,REM,WAKE) [24], as opposed to AASM's 5 stages [2]. To compare the performance to other solutions, the N4 and N3 stages are merged together. Physionet-2018 database has recordings from 994 participants. we re-sample the EEG to 100 Hz, and use EEG channel C3-A2, and put k equal to 5 for cross validation. The DOD has recordings from healthy subjects (DOD-H) and patients with pathological sleep (DOD-O). We used the DOD-H subset using the EEG channel F3-M2. This subset has 25 participants and we used a leave-one-out (LOO) approach for evaluation. The EEG was re-sampled to 100Hz for this database as well. Table II shows the size, setup and signal of interest used for benchmarking MorphuesNet.

Europe PMC Funders Author Manuscripts

IV Results Summary

Table III summarizes the performance in comparison to other models. Models denoted with † can't be compared directly because of use of different subsets for validation, not using single EEG channel etc. Overall, our model achieves accuracy values close to more complex models. As it can be seen, MorphuesNetQ has very close performance to MorphuesNet. However, in Physionet-2018 database, we observed that MorphuesNetQ's performance drop was worse than other databases. We addressed this by not quantizing the start block and the identity block, and this improved performance (indicated by **). After the model is trained, it has a size of a about 300KB. This already can be uploaded on more sophisticated hardware platforms. After further compression, Using quantization to 8-bit integers, the final model size was 50KB. With this size, the model can be uploaded onto a range of commercially available MCUs. To assess embedded deployment, we uploaded the model onto the Arduino Nano 33 BLE development board [25]. A digital pin was used to measure the time required to carry out inference. The latency for one inference was 1.6 seconds.

V Discussion

The model outlined in this paper, MorphuesNet, is a highly compact and optimized model designed for deployment on wearables. It has very few parameters and lower number of arithmetic operations compared to other sleep stage classifiers. As the model is to enable deploying therapies at-home, the model can be fine-tuned to detect certain sleep stages.

As MorphuesNet has much less parameters compared to other models (shown in Table III), one can argue that it can't generalize well to databases with larger patient pools. As it is evident in Table III, MorphuesNet does in fact perform comparable to state of the art.

At its worst, accuracy drops around 4%. MF1, however, shows more drop. This is primarily because of the model's low sensitivity of detecting sleep stage N1. As MorphuesNet only receives the raw EEG signal as its input, it does not receive spectrum based features. We hypothesize that feeding a simple Power Spectral Density (PSD) of the EEG segment would help improve sensitivity. We had empirical findings to confirm this on Sleep-EDF 78 database.

As the model is designed for carrying out real-time SSC for therapies, there are stages that have been targeted for specific therapy protocols. NREM stages 2 and 3, and REM sleep have been explored for TMR. MorphuesNet has comparable sensitivity to these states (minimum of 0.77 in REM) to state of the art. This indicates that it can be used for sleep-based therapies.

VI Future Work

This model will be tested on a headset, designed in-house. This headset will be tested against clinical grade PSG setup for comparison of both signal quality, and algorithm performance. This will be a cross-site validation study. Further potential improvement would be the addition of EOG as well, as it has shown to improve classification performance in certain cases [8].

VII Limitations and Considerations

It is worth mentioning that when it comes to deployment in an at-home environment, deploying a full PSG setup is problematic. One problem, for instance, is making sure the electrodes are making good contact with the scalp. This is challenging specially when dealing with hair. Clinical PSGs use gel-based wet electrodes. However, in at-home scenarios, EEG collection is restricted to the forehead when using wearables. This implies that the deployed SSC algorithm needs to be trained on classifying sleep stage only using EEG collected from forehead.

Acknowledgment

The authors would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work.

References

[1]. Cairney, et al. The Benefits of Targeted Memory Reactivation for Consolidation in Sleep are Contingent on Memory Accuracy and Direct Cue-Memory Associations. Sleep. 2016. May.

[2]. BIber C, et al. The AASM manual for the scoring of sleep and associated events: rules, terminology, and technical specification. American Academy of Sleep Medicine. 2007.

[3]. Antony, JW, Paller, KA. The hippocampus from cells to systems. Springer; Cham: 2017. 245–280.

[4]. Göldi M, van Poppel EAM, Rasch B, Schreiner T. Increased neuronal signatures of targeted memory reactivation during slow-wave up states. Scientific reports. 2019; 9 (1) 1–10. [PubMed: 30626917]

[5]. Navarrete, et al. Examining the optimal timing for closed-loop auditory stimulation of slow-wave sleep in young and older adults. Sleep. 2020; 43 (6) zsz315 [PubMed: 31872860]

[6]. Wassing R, et al. Restless REM Sleep Impedes Overnight Amygdala Adaptation. Curr Biol. 2019; 29: 2351–2358. e4 [PubMed: 31303489]

[7]. Pierrick, et al. The Dreem Headband compared to polysomnography for electroencephalographic signal acquisition and sleep staging. Sleep. 2020; November. 43 (11)

[8]. Phanand H, et al. XSleepNet: Multi-View Sequential Model for Automatic Sleep Staging. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2022; Sept 1; 44 (9) 5903–5915. [PubMed: 33788679]

[9]. Supratak A, Dong H, Wu C, Guo Y. DeepSleepNet: a model for automatic sleep stage scoring based on raw single-channel EEG. IEEE Transactions on Neural Systems and Rehabilitation Engineering. 2017.

[10]. Zhuang L, Dai M, Zhou Y, Sun L. Intelligent automatic sleep staging model based on CNN and LSTM. Front Public Health. 2022. July.

[11]. Xu Z, et al. Sleep stage classification using time-frequency spectra from consecutive multi-time points, Frontiers. Frontiers. 2020. Accessed: April 1, 2023

[12]. Guillot A, Sauvet F, During EH, Thorey V. Dreem Open Datasets: Multi-Scored Sleep Datasets to Compare Human and Automated Sleep Staging. IEEE Transactions on Neural Systems and Rehabilitation Engineering. 2020; September; 28 (9) 1955–1965. [PubMed: 32746326]

[13]. Supratak, A; Guo, Y. TinySleepNet: An Efficient Deep Learning Model for Sleep Stage Scoring based on Raw Single-Channel EEG; 2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC); Canada. 2020. 641–644.

[14]. Thompson NC, Greenewald K, Lee K, Manso GF. The computational limits of Deep Learning. arXiv: 200705558. 2022. July.

[15]. FDA Cybersecurity Safety Communications. Accessed: Apr. 2023 [Online]. Available: https://www.fda.gov/medical-devices/digital-health/cybersecurity

[16]. https://github.com/ali77sina/MorphuesNet

[17]. Liu, Hanxiao; Simonyan, Karen; Yang, Yiming. arxive:180609055; DARTS: Differentiable Architecture Search International Conference on Learning Representations; 2019.

[18]. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. arXiv:151203385. 2015. December.

[19]. Chollet F. Xception: Deep learning with depthwise separable convolutions. arxiv:1610.02357. 2017. April.

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

[21]. Agarap AF. Deep learning using rectified linear units (ReLU. arXiv:180308375. 2019.

[22]. Kemp B, Zwinderman AH, Tuk B, Kamphuisen HAC, Oberyé JJL. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the EEG. IEEE-BME. 2000; 47 (9) 1185–1194.

[23]. Ghassemi, et al. You snooze, you win: the physionet/computing in cardiology challenge; 2018 Computing in Cardiology Conference (CinC); 2018 September. 1–4.

[24]. Moser D, et al. Sleep classification according to AASM and Rechtschaffen & Kales: effects on sleep scoring parameters. Sleep. 2009; Feburary; 32 (2) 139–49. [PubMed: 19238800]

[25]. Kurniawan A. Iot projects with arduino nano 33 ble sense. Berkeley: Apress. 2021; 129

[26]. Phan, Huy; , et al. Towards more accurate automatic sleep staging via deep transfer learning. IEEE Transactions on Biomedical Engineering. 2020.

[27]. Fiorillo L, Favaro P, Faraci FD. DeepSleepNet-Lite: A Simplified Automatic Sleep Stage Scoring Model With Uncertainty Estimates. IEEE Transactions on Neural Systems and Rehabilitation Engineering. 2021; 29: 2076–2085. [PubMed: 34648450]

Europe PMC Funders Author Manuscripts

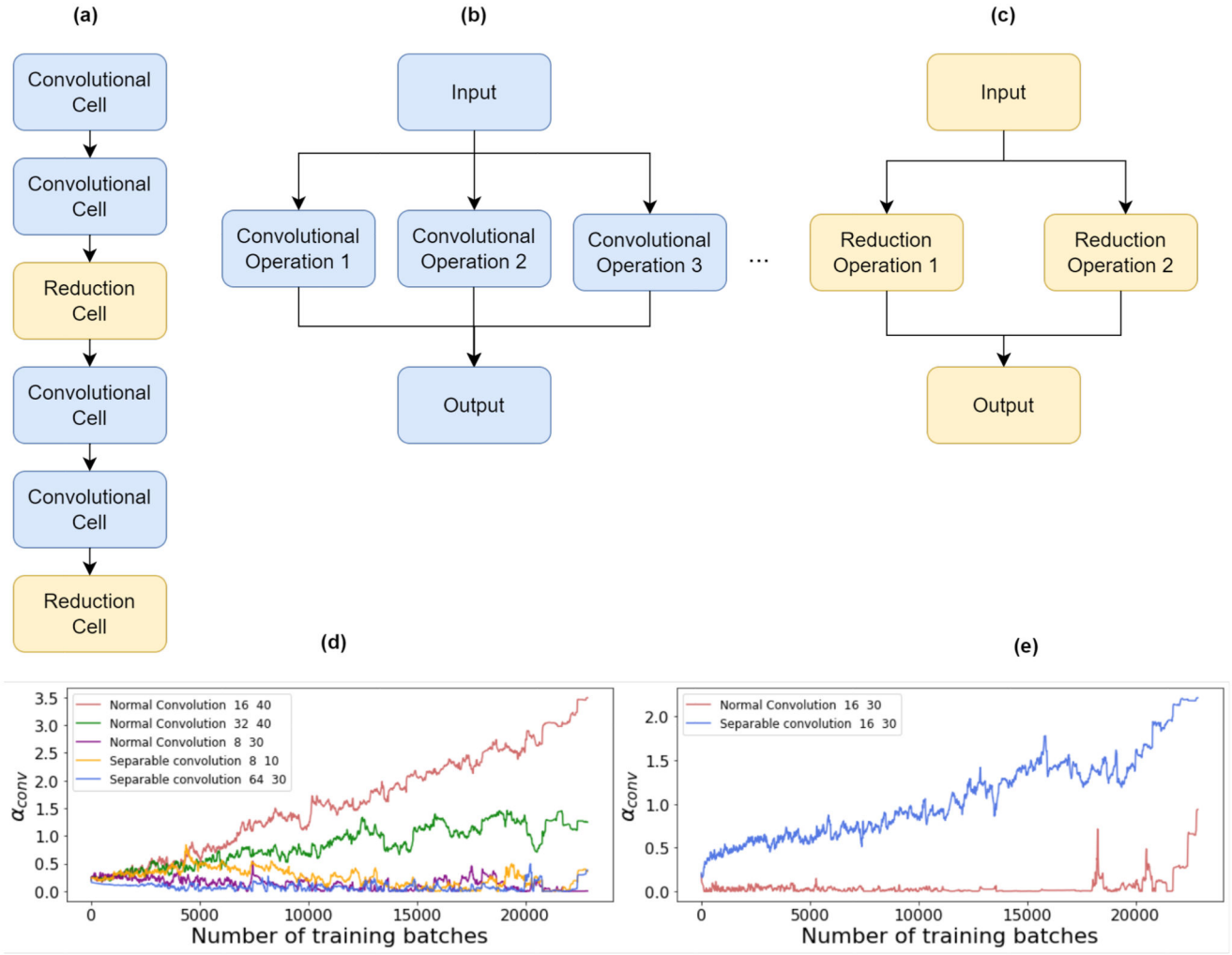

Fig. 1.

(a): overall view of the architecture search model, light blue cells include different convolutional operations with varying filter/kernel sizes, and yellow cells are reduction cells that include either average or max pooling operations. **(b)**: architecture of a convolutional cell, with each edge having an importance factor **(c)**: architecture of a reduction cell. **(d)**: importance factor of different convolutional operations for the first convolutional cell, favoring one of the normal convolutions as the desired operation. **(e)**: Importance factor of convolutional operations in the second cell, showing the desire to choose a separable convolution over a normal convolution with the same parameters e.g. filters and kernel size

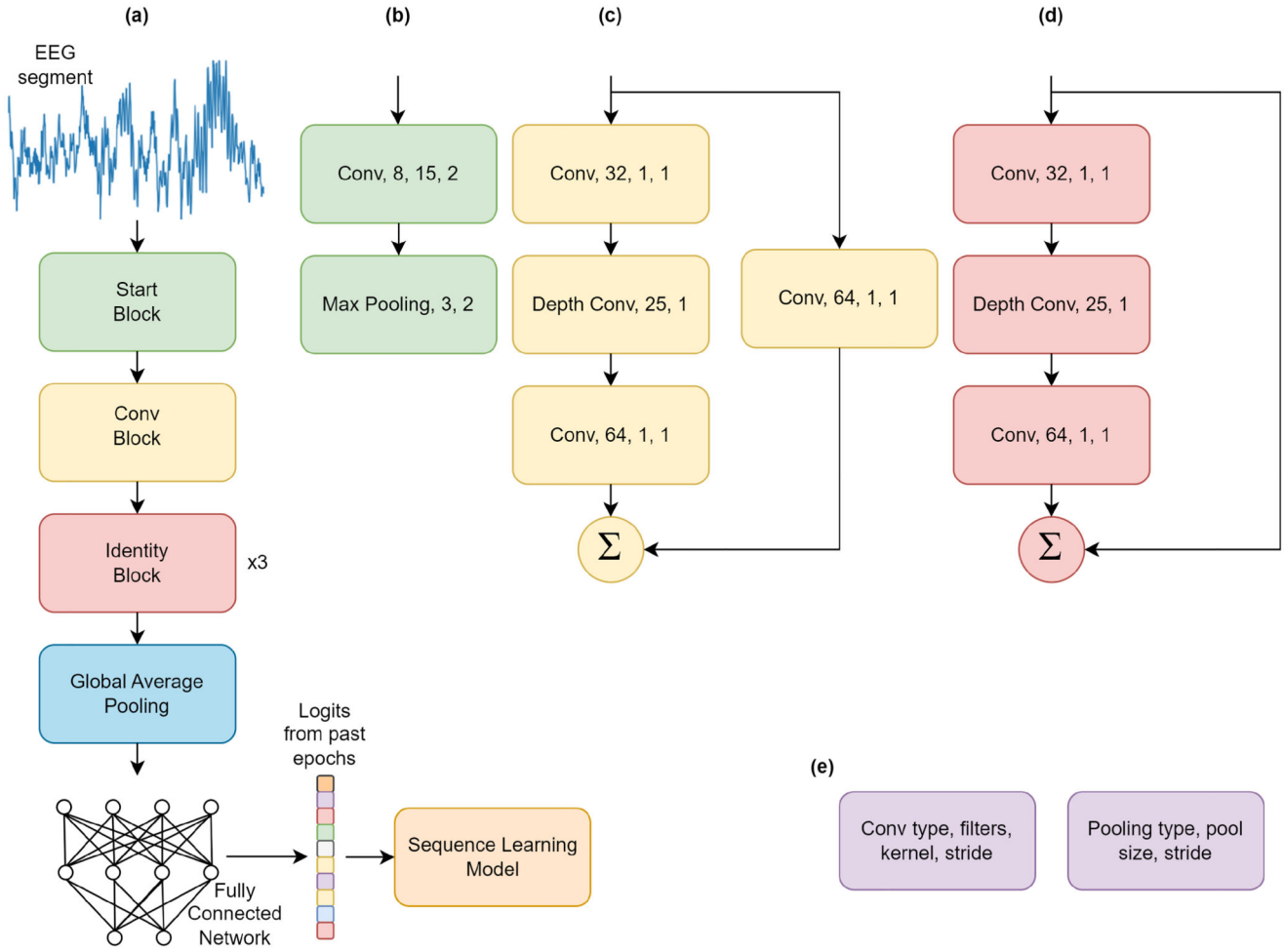

Fig. 2.

(a): Architecture of MorphuesNet. **(b)**: Start block, which includes a normal convolution and a max pooling operation. **(c)**: Conv block, which carries out a depth-wise separable convolution and sums it with a residual connection from the input that goes through a point wise convolution. **(d)**: Identity block, which passes the input tensor through a depth-wise separable convolution and sums it with a residual connection from the input. **(e)**: Convolutional and reduction operation notations used in the figure. Each convolutional block also includes a batch normalization layer before activation.

Europe PMC Funders Author Manuscripts

Europe PMC Funders Author Manuscripts

Table I

Summary Of Requirements

Type	Description	Aim
Real-time	The ability to classify sleep stages in real-time	latency less than 10 seconds for 30 second sleep epochs
Size	Model should be small enough to be deployed on generic MCUs	memory footprint to be less than 100KB
Energy	The device needs to stay operational long enough	Battery to last at least 8 hours
Performance	Whilst being small in size, it should maintain a reasonable performance	comparable sensitivity and specificity to state of the art

Europe PMC Funders Author Manuscripts

Table II

Summary Of Different Databases

Databse	Size	EEG channel	Experimental setup
Sleep-EDF 20	20	Fpz-Cz	20-fold CV
Sleep-EDF 78	78	Fpz-Cz	10-fold CV
DOD-H	25	F3-M2	LOO
Physionet-2018	994	C3-A2	5-fold CV

Table III

Summary Of Performance And Model Size Of Different Models On Different Databases.

Dataset	Model	Parameters(M)	Accuracy	MF1	Sensitivity	Specificity
Sleep-EDF 20 (± 30 min)	MorpheusNet	0.02	84.2	76.3	76.9	95.6
	MorpheusNetQ	0.02	84.7	76.7	77.1	95.8
	XSleepNet2 [8]	5.6	86.3	80.6	80.2	96.4
	TinySleepNet† [13]	1.3	85.4	80.5	-	-
	SeqSleepNet [26]	0.2	85.2	78.4	-	-
	DeepSleepNet-Lite [27]	0.6	84.0	78.0	-	-
Sleep-EDF 20 (in-bed)	MorpheusNet	0.02	81.4	72.0	73.1	94.6
	MorpheusNetQ	0.02	82.1	73.9	74.7	95.0
	XSleepNet2 [8]	5.6	83.9	78.7	78.6	95.2
	FT-XSleepNet† [8]	5.6	85.7	80.8	80.6	96.0
	DeepSleepNet [8] [9]	20	80.8	74.2	-	-
Sleep-EDF 78 (± 30 min)	MorpheusNet	0.02	79.3	72.1	71.6	94.2
	MorpheusNetQ	0.02	80.6	74.1	73.5	94.6
	XSleepNet2 [8]	5.6	84	77.9	77.5	95.7
	SeqSleepNet [8] [26]	0.2	82.6	76.4	-	-
Sleep-EDF 78 (in-bed)	MorpheusNet	0.02	77.8	71.8	70.7	93.8
	MorpheusNetQ	0.02	78.2	72.7	71.8	93.8
	XSleepNet2 [8]	5.6	80.3	76.4	76.1	94.6
	DeepSleepNet [8] [9]	20	78.5	75.3	75.0	94.1
DOD-H	MorpheusNet	0.02	83.8	73.7	76.5	95.2
	MorpheusNetQ	0.02	83.8	75.0	76.4	95.2
	SimpleSleepNet (multi-channel)† [12]	0.094	89.9	82.4	80.2	96.4
	SimpleSleepNet (single-channel)† [12]	0.094	86.7	-	-	-
	Scorer (average)† [12]	-	86.5	78.56	-	-
	Scorer (best)† [12]	-	88.7	80.4	-	-
	Scorer (worst)† [12]	-	81.7	73.7	-	-
	DeepSleepNet† [9] [12]	20	89.6	81.8	-	-
Physionet-2018	MorpheusNet	0.02	78.6	76.0	75.6	94.0
	MorpheusNetQ	0.02	70.0	67.7	67.2	86.5
	MorpheusNetQ**	0.02	77.3	74.4	74.3	93.7
	XSleepNet2† [8]	5.6	80.3	78.6	78.7	94.6