A deep learning architecture to detect events in EEG signals during sleep

Stanislas Chambon, Valentin Thorey, Pierrick J Arnal, Emmanuel Mignot, Alexandre Gramfort

A DEEP LEARNING ARCHITECTURE TO DETECT EVENTS IN EEG SIGNALS DURING SLEEP

Stanislas Chambon[1,2,3*], *Valentin Thorey*[2*], *Pierrick J. Arnal*[2], *Emmanuel Mignot*[1], *Alexandre Gramfort*[3,4,5]

[1] Center for Sleep Sciences and Medicine, Stanford University, Stanford, California, USA
[2] Research & Algorithms Team, Dreem, Paris, France
[3] LTCI Télécom ParisTech, Université Paris-Saclay, Paris, France
[4] Inria, Université Paris-Saclay, Paris, France
[5] CEA Neurospin, Université Paris-Saclay, Paris, France

ABSTRACT

Electroencephalography (EEG) during sleep is used by clinicians to evaluate various neurological disorders. In sleep medicine, it is relevant to detect macro-events (≥ 10 s) such as sleep stages, and micro-events (≤ 2 s) such as spindles and K-complexes. Annotations of such events require a trained sleep expert, a time consuming and tedious process with a large inter-scorer variability. Automatic algorithms have been developed to detect various types of events but these are event-specific. We propose a deep learning method that jointly predicts locations, durations and types of events in EEG time series. It relies on a convolutional neural network that builds a feature representation from raw EEG signals. Numerical experiments demonstrate efficiency of this new approach on various event detection tasks compared to current state-of-the-art, event specific, algorithms.

***Index Terms*—** Deep learning, EEG, event detection, sleep, EEG-patterns, time series

1. INTRODUCTION

During sleep, brain activity is characterized by some specific electroencephalographic (EEG) patterns, or events, (e.g. spindles, K-complexes) used to define more global state (e.g. sleep stages) [1]. Detecting such events is meaningful to better understand sleep physiology [2, 3] and relevant to the physiopathology of some sleep disorders [4, 5, 6]. Traditionally, visual analysis of these events is conducted, but it is tedious, time consuming and requires a trained sleep expert. Agreement between experts is often low, although this can be improved by taking consensus of multiple sleep experts [2].

Automatic detection algorithms have been proposed to detect specific types of micro-events in sleep EEG. These typically build on a band-pass filtering step within a certain fixed frequency band, for instance $11 - 16$ Hz for the detection of spindles. Three types of methods can be distinguished. The first type relies on extracting the envelop of the filtered signal and performing a thresholding-like step with either a fixed or tunable threshold [7, 8, 9, 10, 11, 12]. It is primarily used for spindle detection. The main difference between these lies in the thresholding part which is either performed on the rectified filtered signal [12], or on the instantaneous amplitude obtained after an Hilbert-Transform [11], on the root mean square of the filtered signal [10] or on the moving average of the rectified filtered signal [8]. This first type of methods can identify start and end times of events by looking at inflexion points of the envelop of the filtered signal [9]. Furthermore, most of these approaches are specific to a sleep stage [12, 11, 10, 8, 7] and rely heavily on preprocessing steps such as notch filtering around 50 Hz to remove the electrical current artefacts or general visual artefacts removal, impractical on large amounts of data. The second type of methods relies on decomposing the EEG into its oscillatory and transient components. Then filtering and thresholding are used to detect events of interest [13, 14, 15]. These are more general methods since they can detect either sleep spindles or K-complexes and can work over entire sleep recordings independently of sleep stages. The third type of methods corresponds to machine learning methods and are more general. These filter the EEG signal, extract spectral and temporal features from a segment and predict with a binary SVM whether it is a spindle [16]. These methods all have significant limitations: (1) use of band-pass filtering with fixed cut-off frequencies that might not be adapted to some subjects (2) they are intrinsically event specific (3) their hyper parameters are often optimized on the record used for evaluating the performance, introducing an overfitting bias in reported results.

A solution to these problems may be found in the adaptation of recent computer vision algorithms developed in the context of object detection. Indeed, detecting events in multivariate EEG signals consist in predicting time locations, dura-

*Contributed equally. This work was supported in part by the french Association Nationale de la Recherche et de la Technologie (ANRT) under Grant 2015 / 1005

tion and types of events which is closely related to the object detection problem in computer vision where bounding boxes and objects categories are to be predicted. For this latter problem, state-of-the-art methods make use of dedicated deep neural networks architectures [17, 18, 19, 20].

In this paper, we introduce such a dedicated neural network architecture to detect any type of event over the sleep EEG signal. The proposed approach builds on a convolutional neural network which extracts high-level features from the raw input signal(s) for multiple default event locations. A localization module predicts an adjustment on start and end times for those default event locations while a classification module predicts their labels. The whole network architecture is trained end-to-end by back-propagation. In this paper, we first present the general method and the architecture of the proposed neural network. We then evaluate performance of the proposed approach versus the current state-of-the-art methods on 2 event detection tasks.

Notation We denote by $[\![n]\!]$ the set $\{1, \dots, n\}$ for $n \in \mathbb{N}$. Let $\mathcal{X} \in \mathbb{R}^{C \times T}$ be the set of input signals, C is the number of EEG channels and T the number of time steps. $\mathcal{L} = [\![|L|]\!]$ stands for labels of events where L is the number of different labels. 0 is the label associated to no event or background signal. An event $e = \left\{t^c, t^d, l\right\} \in \mathcal{E} = \mathbb{R}^2 \times \mathcal{L} \cup \{0\}$ is defined by a center location time t^c, a duration t^d and an event label (or type) $l \in \mathcal{L}$. A *true event* is an event with label in $\mathcal{L}$ detected by a human scorer or a group of human scorers, *a.k.a. gold standard*. A *predicted event* is an event with label in $\mathcal{L}$ detected by an algorithm or a group of algorithms.

2. METHOD

The detection procedure employed by our predictive system is illustrated in Figure 1 (with C=1). Let $x \in \mathcal{X}$ be an input EEG signal. First, default events are generated over the input signal, *e.g.* 1 s events every 0.5 s if this corresponds to a typical duration of events to be detected (cf. Figure 1-A). Positions, durations and overlaps of default events can be modified. Then, the network predicts for each default event its adjusted center and duration, together with the label of the potential event (cf. Figure 1-B). Events with the highest probabilities are selected, and non-maximum suppression is applied to remove overlapping predictions (cf. Figure 1-C). This is similar to the SSD [18] and YOLO approaches [19] developed in the context of object detection. Training this predictive system requires the following steps. First, default events are generated over the input signal and matched to the true events based on their Jaccard index, *a.k.a.* Intersection over Union (IoU) [19]. The network is trained to predict the centers and durations together with the labels of the events. Default events which do not match a true event with a sufficient IoU are assigned the label $l = 0$. To address the issue of label imbalance between real events and events with label $l = 0$, subsampling is used so that only a fraction of events with label $l = 0$ is used for training.

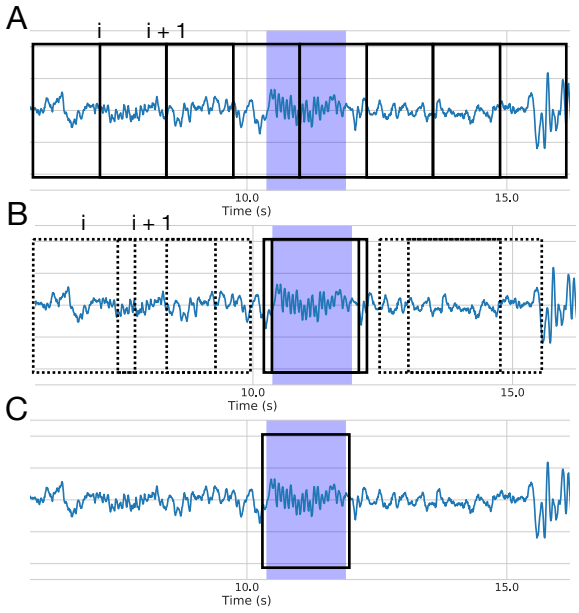

Fig. 1. Proposed approach prediction procedure inspired by SSD [18]: A: default events are generated over the input signals. B: the network predicts refined locations of any default event and its label included the label no event: 0. C: non-suppression maximum is applied to merge overlapping events with label different from 0. The network finally returns the locations of the merged events and their labels

In order to learn a system to achieve the prediction task just described with back-propagation, one needs to design a fully differentiable architecture. The aim is to learn a function $\hat{f}$ from $\mathcal{X}$ to $\mathcal{Y}$ where $y \in \mathcal{Y}$ is a set of elements from $\mathcal{E}$. Let N_d be the number of default events generated over the input signal $x \in \mathcal{X}$. It is also the number of adjusted events predicted by the network. Let $D(x) = \left\{d_i = (t_i^c, t_i^d), i \in [\![N_d]\!]\right\}$ be the set of centers and durations of the N_d default events generated over x. Let $E(x) = \left\{e_j = (t_j^c, t_j^d, l_j) : j \in [\![N_e]\!]\right\}$ be the list of the N_e true events annotated over the signal x. Default events which match a true event are selected to train the localization and classification capacities of the system. The $\mathrm{IoU}(d_i, e_j) \in [0, 1]$ is computed between each default event $d_i \in D(x)$ and each true event from $e_j \in E(x)$. Let $\eta > 0$, d_i matches e_j if $\mathrm{IoU}(d_i, e_j) \geq \eta$. If multiple true events match the same default event d_i, the true event which exhibits the highest IoU with d_i is selected. We introduce the function γ which returns, if it exists, the index of the true event matching with the default event d_i, and $\emptyset$ otherwise:

$$\gamma(i) = \underset{\substack{j \in [\![N_e]\!] \\ \mathrm{IoU}(d_i, e_j) \geq \eta}}{\arg\max} \ \mathrm{IoU}(d_i, e_j) \in [\![N_e]\!] \cup \{\emptyset\}$$

Let d_i be a default event matching with the true event e_j.

d_i's center and duration are then encoded with $\phi_{e_j} : \mathbb{R}^2 \longrightarrow \mathbb{R}^2, d_i = (t_i^c, t_i^d) \longmapsto \left(\frac{t_j^c - t_i^c}{t_i^d}, \log \frac{t_j^d}{t_i^d} \right)$ [20]. This encoding function quantifies the relative variations in centers and durations between the default event d_i and the true event e_j, and represents the quantities the network actually predicts. Let $\hat{f}(x) \in \mathcal{Y}$ be the prediction made by model $\hat{f}$ over x. We define it as $\hat{f}(x) = \{(\hat{t}_i^c, \hat{t}_i^d, \hat{l}_i) \in \mathcal{E}, i \in [\![N_d]\!]\}$. $\hat{\tau}_i = (\hat{t}_i^c, \hat{t}_i^d)$ are the predicted coordinates of encoded default event d_i and $\hat{l}_i$ is its predicted label. In practice, the model will output the probability of each label $l \in \mathcal{L} \cup \{0\}$ for default event d_i so $\hat{l}_i$ is replaced by $\hat{\pi}_i \in [0,1]^{|\mathcal{L}|+1}$. As it is a probability vector, we have $\sum_{l \in \mathcal{L} \cup \{0\}} \hat{\pi}_i^l = 1$.

The loss between the true annotation $E(x)$ and the model prediction $\hat{f}(x)$ over signal x is a function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ defined as $\ell\left(E(x), \hat{f}(x)\right) = \ell^+_{norm} + \ell^-_{norm}$ where

$$\ell^+ = \sum_{\substack{i \in [\![N_d]\!] \\ \gamma(i) \neq \emptyset}} \mathrm{L1}_{smooth}\left(\phi_{e_{\gamma(i)}}(d_i), \hat{\tau}_i\right) - \log(\hat{\pi}_i^{l_{\gamma(i)}}) \quad (1)$$

$$\ell^- = - \sum_{\substack{i \in [\![N_d]\!] \\ \forall j \in [\![N_e]\!] : \mathrm{IoU}(d_i, e_j) < \eta}} \log(\hat{\pi}_i^0) \quad (2)$$

ℓ^+_{norm} (resp. ℓ^-_{norm}) is obtained by dividing ℓ^+ (resp. ℓ^-) by the number of terms in the sum (1) (resp. (2)). In (1), we sum the localization and classification loss for any default event d_i matching a true event $e_{\gamma(i)}$. The $\mathrm{L1}_{smooth}$ loss applies coordinate-wise the real valued function: $x \mapsto (x^2/2)\mathbb{1}_{|x|<1} + (|x| - 1/2)\mathbb{1}_{|x| \geq 1}$ [20]. Equation (2) stands for the classification loss of default boxes which do not match any true event. Subsampling has been ommitted in (2) for simplicity. In practice, we use a $1/3$ ratio between default events matching true events and those not matching a true event, selecting those with the worst classification scores.

In the end, the learning problem consists in solving the following minimization problem to obtain event detector $\hat{f}$:

$$\hat{f} \in \arg\min_{f \in \mathcal{F}} \mathbb{E}_{x \in \mathcal{X}} \left[\ell\left(E(x), f(x)\right)\right] \quad (3)$$

In order to specify what is the function class $\mathcal{F}$, one needs to detail the network architecture. We consider a general convolutional network that, given a set of default events $D(x) = \{d_i = (2^8 \cdot (i - 0.5)/\rho,\ t_i^d) : i \in [\![N_d]\!]\}$, predicts N_d events, where $N_d = T \cdot \rho / 2^8$ and $\rho \in \mathbb{N}$ is an overlapping factor. The network is composed of 3 parts, see Table 1.

The first part, called Block 0, performs a spatial filtering in order to increase the Signal to Noise Ratio (SNR) by recombining the original EEG channels into virtual channels using a 2D spatial convolution [21]. It takes as input a tensor $x \in \mathcal{X}$ and outputs a new tensor $x_0 \in \mathcal{X}$. Channels of x_0 are obtained by linear combination of the channels of x. It can be seen as a 2D convolution with C kernel of size (C, 1), a stride of 1 and a linear activation. It is followed by a transposition to permute the channel and spatial dimensions in order to recover a tensor $x_0 \in \mathcal{X}$. If $C = 1$, this block is skipped.

The second part, composed of Block k, for $k \in [\![8]\!]$, performs feature extraction over x_0 in the time domain. Each block is composed of a 2D convolution layer with batch normalization [22] and ReLU activation $x \mapsto \max(x, 0)$ [23], followed by a temporal max-pooling. Block k first convolves the previous feature maps x_{k-1} with 4×2^k kernels of size $(1, 3)$ (space, time), using a stride of 1. Zero padding is used to maintain the dimension of the tensor through the convolution layer. Then, the ReLU activation is applied. Finally a temporal max pooling operation with kernel of size $(1, 2)$ and stride 2 is applied to divide by 2 the temporal dimension. Each Block k does not process the spatial dimension.

The third part, Block 9 takes as input a tensor $x_8 \in \mathbb{R}^{1024 \times C \times T/2^8}$, and for each default event $i \in [\![N_d]\!]$, it predicts the event label and its encoded center and duration. Block 9 has two layers: a classification layer 9-a and a localization layer 9-b. Layer 9-a convolves the last feature maps with $(|\mathcal{L}| + 1) \times \rho$ kernels of size $(C, 3)$ (stride of 1). A *softmax* operation is applied along the channel dimension on every set of $(|\mathcal{L}| + 1)$ channels so that, for each default event d_i, one obtains the probability $\hat{\pi}_i^l$ of the corresponding event i to belong to any of the classes $l \in \mathcal{L} \cup \{0\}$. Similarly, layer 9-b convolves the last feature maps x_8 with $2 \times \rho$ kernels of size $(C, 3)$ and stride 1. This gives the predicted coordinates $\hat{\tau}_i = (\hat{t}_i^c, \hat{t}_i^d)$ of any encoded default event $i \in [\![N_d]\!]$.

3. EXPERIMENTS

Data The experiments were performed on MASS SS2 [24]: 19 records from 19 subjects (11 females, 8 males, $\sim$ 23.6 $\pm$ 3.7 years old), sampled at 256 Hz. The spindles have been scored by expert E1 (resp. E2) over 19 records (resp. 15) using different guidelines [14] resulting in $\sim$ 550 (resp. $\sim$ 1100) scored spindles per record. For records scored by both E1 and E2 $\sim$ 500 spindles per record exhibit IoU > 0 (Gaussian-like distribution, pic at 0.6). The 15 records annotated by E1 and E2 were used for spindles detection benchmark. For K-complex detection, and joint spindle and K-complex detection, the 19 records scored by E1 were used.

Cross validation A 5 split cross validation was used. A split stands for 10 training, 2 validation and 3 (resp. 4) testing records for spindle detection (resp. K-complex and joint spindle and K-complex detection).

Metrics *By event metrics* [2] were used to quantify the detection and localization performances of detectors. They rely on an IoU criterion: for a given $\delta > 0$, a predicted event was considered as a true positive if it exhibited an IoU $\geq \delta$ with a true event otherwise it was considered as a false positive.

	Layer	Layer Type	# kernels	output dimension	activation	kernel size	stride
Block 0	1	Convolution 2D	C	(C, 1, T)	linear	(C, 1)	1
	2	Transpose	-	(1, C, T)	-		-
Block k for $k \in [\![8]\!]$	k-a	Convolution 2D	4×2^k	$(4 \times 2^k, C, T/2^{k-1})$	ReLU	(1, 3)	1
	k-b	Max Pooling 2D	-	$(4 \times 2^k, C, T/2^k)$	-	(1, 2)	2
Block 9	9-a	Convolution 2D	$(\lvert\mathcal{L}\rvert + 1) \times \rho$	$(\lvert\mathcal{L}\rvert + 1) \times \rho, 1, T/2^8)$	softmax (channel dimension)	(C, 3)	1
	9-b	Convolution 2D	$2 \times \rho$	$(2 \times \rho, 1, T/2^8)$	linear	(C, 3)	1

Table 1. Model architecture: Block 0 performs spatial filtering and outputs a tensor $\mathbf{x}_0 \in \mathcal{X}$. Block k, $k \in [\![8]\!]$, extracts temporal features and outputs a tensor $x_k \in \mathbb{R}^{(4\times 2^k)\times C\times T/2^k}$. Block 9-a performs the classification of any potential event $i \in [\![N_d]\!]$ and Block 9-b predicts the encoded center and duration of this event. Each convolution is followed by a zero padding layer and batch normalization. Note that the batch dimension is omitted. We have $\rho = N_d/(T/2^8)$.

The numbers of positives and true positives were evaluated to compute the precision, the recall and the F1 scores of detectors. Evaluation was performed on entire testing records. Performances reported were averaged over testing records.

Baselines For spindles detection, 8 baselines were benchmarked: *Ferrarelli et al. 2007* [12], *Mölle et al. 2011* [10], *Nir et al. 2011* [11], *Wamsley et al. 2012* [8], *Ray et al. 2015* [7] (Python package `https://github.com/wonambi-python/wonambi`), *Wendt et al. 2012* [9], *Parekh et al. 2017* [13] and *Lajnef et al. 2017* [14]. We used the implementations by the authors of [2, 13, 14]. For K-complex detection, *Lajnef et al. 2017* [14] was compared.

Signal from C3 channel was used by all the baselines except *Parekh et al. 2017* which used: F3, F4, Fz, C3, C4, Cz channels. Hyper-parameters of *Parekh et al. 2017* and *Lajnef et al. 2017* were selected in the ranges provided by the authors of [13, 14] by grid search on the validation data.

Proposed approach The proposed approach was benchmarked on signal from channel C3. The network was provided with 20 s samples $x \in \mathbb{R}^{C\times T}$ with $C = 1$ and $T = 5120$ ($256\,Hz$ sampling). A normalization was applied to x: centering and standardization by dividing each centered signal by its standard deviation computed on the full record.

The approach was implemented with PyTorch library [25]. Minimizing (3) was achieved with stochastic gradient descent using a learning rate $\mathrm{lr} = 10^{-3}$, a momentum $\mu = 0.9$ and a batch size of 32. 100 training epochs were considered. Each sample was randomly selected to contain at least a true event. When a true event was partially included in a sample, its label l was set to 0 if less than 50% of this event was part of that sample. Early stopping was used to stop the training process when no improvement was observed on the loss evaluated on validation data after 5 consecutive epochs.

Matching hyper-parameter η was fixed to $\eta = 0.5$. Default event hyper-parameters were fixed to: $\rho = 4$, $N_d = 80$, $t_i^d = 256$. The resulting set $D(x) = \{(64 \cdot i - 32, 256) : i \in [\![N_d]\!]\}$ seemed a reasonable choice given the fact that both spindles and K-complexes have ~ 1 s duration. A potential event i was considered as a positive event of label $l \in \mathcal{L}$ if $\hat{\pi}_i^l \geq \theta^l$. Hyper parameter θ^l was selected by grid search over the validation data.

Spindles Detectors were compared to 4 gold standards: events scored by E1, E2, the union and the intersection of events scored by both E1 and E2, see Figure 2.

First, the proposed approach seems to detect the occurrence of spindles without any supplementary information regarding the sleep stages but also to localize the spindles accurately. Indeed, the proposed approach outperforms the baselines in terms of precision / recall at $\mathrm{IoU} = 0.3$ and exhibits an higher F1 than the baselines for any IoU. Second, the proposed approach seems to take into account any considered gold standard. Indeed, it exhibits stable performances over the gold standards contrarily to most of the baselines, except *Parekh et al. 2017* and *Lajnef et al. 2017*.

K-complexes Performances are reported in Figure 3. The proposed approach seems to outperform the baseline in terms of precision / recall at $\mathrm{IoU} = 0.3$ and exhibits a higher F1 score than the compared baseline at any IoU.

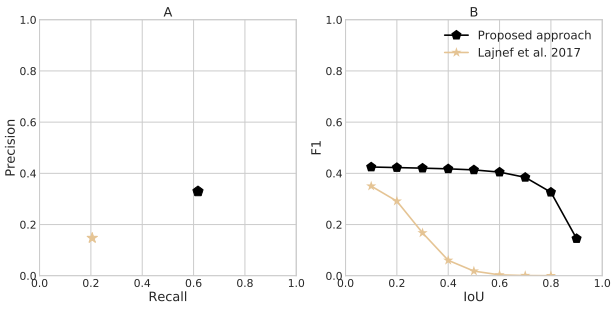

Fig. 3. K complex detection. A: precision / recall at $\mathrm{IoU} = 0.3$. B: F1 score as a function of IoU.

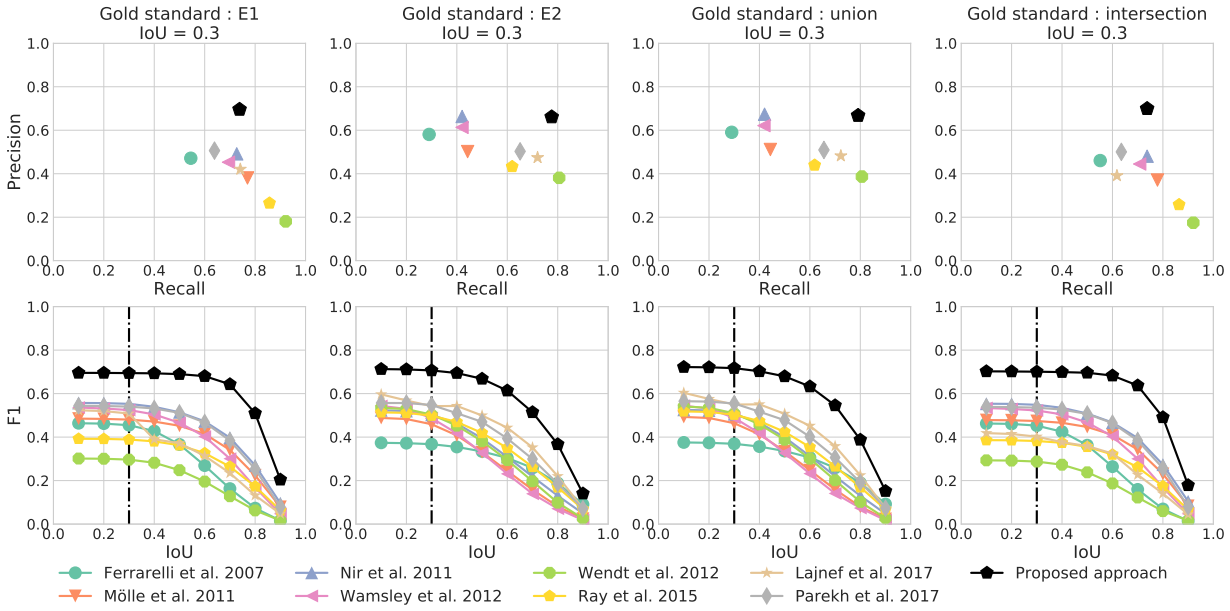

Fig. 2. Spindle detection: benchmark with respect to 4 gold standards: the proposed approach outperforms the baselines. First row: averaged precision / recall at IoU $= 0.3$. Second row: F1 score as a function of IoU

Detecting events jointly or separately The proposed approach was trained to detect both spindle and K-complex jointly and separately. Performances are reported in Figure 4. Same performances are obtained when the method is trained to detect spindles and K-complexes jointly or separately.

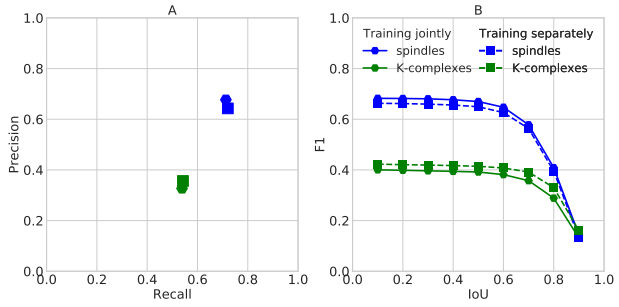

Fig. 4. Detecting spindles and K-complexes jointly or separately leads to similar performances. A: Precision / Recall of detectors at IoU = 0.3. B: F1 scores as a function of IoU.

4. DISCUSSION

The proposed approach builds on deep learning to learn a feature representation relevant for detecting any type of event. Surprisingly enough, the approach handles well the task of detecting spindles and K-complexes using *only* 10 training and 2 validation records. We performed additional experiments (not shown) to vary the number of training records from 1 to 10: the method works when only 1 training record is available. This might be due to random sampling which performs a kind of data augmentation.

As the proposed approach can handle multiple channels, additional experiments on spindles / K-complex detection were run using multiple channels: F3, F4, C3, C4, O1, O2. This did not result in any significant gain of performance on the used dataset (not shown). The method can also handle multiple modalities, electromyography (EMG), electrooculography (EOG) or breathing, and mutiple default event scales at the same time, a property that was not explored in this study but that may be critical for detecting other types of events. This will be addressed in future studies.

The proposed approach seems to perform quite well with respect to different gold standards. Yet it remains to study how the method performs compared to the inter-scorer agreement [2]. This shall be also addressed in future works.

5. CONCLUSION

This paper introduces a new deep learning architecture that can perform event detection of any type over an entire night of EEG recording. The proposed approach learns by back-propagation to build a feature representation from the raw input signal(s), and to predict both locations, durations and types of events. Numerical experiments on spindles and K-complexes detection demonstrate that the proposed approach outperforms previously published detection methods. A major advantage of the proposed approach is that it can detect jointly multiple types of events.

6. REFERENCES

[1] C. Iber, S. Ancoli-Israel, A. Chesson *et al.*, *The AASM Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and Technical Specifications*. American Academy of Sleep Medicine, 2007.

[2] S. C. Warby, S. L. Wendt, P. Welinder *et al.*, "Sleep spindle detection: crowdsourcing and evaluating performance of experts, non-experts, and automated methods," *Nature methods*, vol. 11, no. 4, pp. 385–392, 2014.

[3] S. M. Purcell, D. S. Manoach, C. Demanuele *et al.*, "Characterizing sleep spindles in 11,630 individuals from the National Sleep Research Resource," *Nature Communications*, vol. 8, p. 15930, jun 2017.

[4] J. B. Stephansen, A. Ambati, E. B. Leary *et al.*, "The use of neural networks in the analysis of sleep stages and the diagnosis of narcolepsy," *arXiv:1710.02094*, 2017.

[5] D. S. Manoach, J. Q. Pan, S. M. Purcell *et al.*, "Reduced sleep spindles in schizophrenia: A treatable endophenotype that links risk genes to impaired cognition?" *Biological psychiatry*, vol. 80, no. 8, pp. 599–608, oct 2016.

[6] E. S. Musiek, D. D. Xiong, and D. M. Holtzman, "Sleep, circadian rhythms, and the pathogenesis of Alzheimer Disease," *Exp Mol Med*, vol. 47, no. 3, mar 2015.

[7] L. B. Ray, S. Sockeel, M. Soon *et al.*, "Expert and crowd-sourced validation of an individualized sleep spindle detection method employing complex demodulation and individualized normalization," *Frontiers in Human Neuroscience*, vol. 9, p. 507, sep 2015.

[8] E. J. Wamsley, M. A. Tucker, A. K. Shinn *et al.*, "Reduced sleep spindles and spindle coherence in schizophrenia: Mechanisms of impaired memory consolidation?" *Biological psychiatry*, vol. 71, no. 2, pp. 154–161, jan 2012.

[9] S. L. Wendt, J. A. E. Christensen, J. Kempfner *et al.*, "Validation of a novel automatic sleep spindle detector with high performance during sleep in middle aged subjects," in *Proc. IEEE EMBC*, 2012, pp. 4250–4253.

[10] M. Mölle, T. O. Bergmann, L. Marshall *et al.*, "Fast and Slow Spindles during the Sleep Slow Oscillation: Disparate Coalescence and Engagement in Memory Processing," *Sleep*, vol. 34, no. 10, pp. 1411–1421, 2011.

[11] Y. Nir, R. Staba, T. Andrillon *et al.*, "Regional Slow Waves and Spindles in Human Sleep," *Neuron*, vol. 70, no. 1, pp. 153–169, apr 2011.

[12] F. Ferrarelli, R. Huber, M. J. Peterson *et al.*, "Reduced sleep spindle activity in schizophrenia patients," *Am J Psychiatry*, vol. 164, no. 3, pp. 483–492, 2007.

[13] A. Parekh, I. W. Selesnick, R. S. Osorio *et al.*, "Multichannel sleep spindle detection using sparse low-rank optimization," *J. Neurosci. Methods*, vol. 288, pp. 1–16, 2017.

[14] T. Lajnef, C. O'Reilly, E. Combrisson *et al.*, "Meet Spinky: An Open-Source Spindle and K-Complex Detection Toolbox Validated on the Open-Access Montreal Archive of Sleep Studies (MASS)," *Frontiers in Neuroinformatics*, vol. 11, p. 15, mar 2017.

[15] A. Parekh, I. W. Selesnick, D. M. Rapoport *et al.*, "Detection of K-complexes and sleep spindles (DETOKS) using sparse optimization," *J. Neurosci. Methods*, vol. 251, pp. 37–46, 2015.

[16] D. Lachner-Piza, N. Epitashvili, A. Schulze-Bonhage *et al.*, "A single channel sleep-spindle detector based on multivariate classification of EEG epochs: MUSSDET," *J. Neurosci. Methods*, vol. 297, pp. 31–43, 2018.

[17] T. Lin, P. Goyal, R. B. Girshick *et al.*, "Focal loss for dense object detection," *arXiv:1708.02002*, Aug. 2017.

[18] W. Liu, D. Anguelov, D. Erhan *et al.*, "SSD: single shot multibox detector," *arXiv:1512.02325*, Dec. 2015.

[19] J. Redmon, S. Divvala, R. Girshick *et al.*, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. CVPR*, 2016, pp. 779–788.

[20] S. Ren, K. He, R. Girshick *et al.*, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. NIPS*, 2015, pp. 91–99.

[21] S. Chambon, M. N. Galtier, P. J. Arnal *et al.*, "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, 2018.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015, pp. 448–456.

[23] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proc. ICML*, 2010, pp. 807–814.

[24] C. O'Reilly, N. Gosselin, J. Carrier *et al.*, "Montreal Archive of Sleep Studies: an openaccess resource for instrument benchmarking and exploratory research," *J Sleep Res*, vol. 23, no. 6, pp. 628–635, jun 2014.

[25] A. Paszke, S. Gross, S. Chintala *et al.*, "Automatic differentiation in PyTorch," in *NIPS Workshop*, 2017.