Predicting Recovery from Coma After Cardiac Arrest Using Low-level Features from EEG Recordings and a Small-sized LSTM Network

Benjamin Cauchi[1], Marco Eichelberg[1], Andreas Hein[1,2]

[1]OFFIS e.V., Institute for Information Technology, Oldenburg, Germany
[2]Carl von Ossietzky Universität, Oldenburg, Germany

Abstract

This paper presents a computational approach proposed as part of the George B. Moody PhysioNet Challenge 2023. This method uses electroencephalograms (EEGs) to predict the neurological recovery of patients following cardiac arrest and was proposed by the team Oldenburg. It relies on computing a set of low-level time-varying features from a small subset of the available EEG channels. The resulting sequence of features are input to a recurrent neural network based on long short-term memory (LSTM) cells that estimates the probability of the target patient to be in a given cerebral performance category (CPC) status. The resulting model received a Challenge score of 0.025 (ranked 36th out of 36 submissions) on the hidden test set, failing to produce satisfying detection with a false positive rate inferior to 0.05, as required by the Challenge score. However, results on the validation set suggest that the proposed method could yield useful results, though more thorough tuning and evaluation would be needed.

1. Introduction

This paper describes a method submitted to the 2023 George B. Moody PhysioNet Challenge. The aim of this challenge was to to develop automated methods able to predict the outcome of patients being comatose after a cardiac arrest, potentially helping clinicians to predict the neurological recovery of patients. The developed methods were to use long-term electroencephalogram (EEG) recordings, along with additional signals and patient information as input [1, 2]. All development, training and testing were done using the data provided as part of the challenge [3].

The development decision of the proposed method was guided by two main objectives. First, the desire to explore the feasibility of a classification based only on a small number of electrodes and on a small-sized model. This could facilitate the development of compact sensors and be useful for a wide range of applications. Second, and more pragmatic, the necessity for this approach to be implemented and tested using the limited time and computing resources available within the challenge's timeframe.

Within these objectives, the proposed method only use EEG signals as input, disregarding both the other recorded vital parameters (e.g., electrocardiogram) and patients information (e.g, age). Additionally, only a small subset of the available channels of the recorded EEG signals are used as input. These signals are divided into overlapping frames from which low-level features are extracted. These features, and the parameters used to compute them, were chosen for their proven usefulness in previous vital-parameter classification tasks [4, 5]. Aiming at predicting the probability of each cerebral performance category (CPC) status for a given patient, the resulting sequence of feature vectors are input to a recurrent neural network (RNN) of long short-term memory (LSTM) cells, that can take the time dependency of these features into account. The hyper-parameters of this network were set according to previous use on various applications, namely different types of regression from audio signals [6, 7]. This choice was motivated by the need to be applicable to time-variant data while being robust against missing data. The predicted status of the patient and its CPC scores is computed from the probabilities that are estimated by the network.

The remainder of this paper is structured as follows. First, the proposed method is described in Section 2. The experimental setup, detailing the used settings, as well as the obtained results are presented in Section 3. Discussion on the method and results are presented in Section 4.

2. Proposed approach

2.1. Feature extraction

The proposed method computes features from the signal $y_m(n)$ of length N and sampling frequency f_s, where m and n denote the channel and sample index, respectively. This signal is obtained by filtering, resampling and averaging the input EEG. The filtering consists of a bandpass filter with cut-off frequencies b_{low} and b_{high}, and of a zerophase notch filter centred around the utility frequency b_{util}. It can be noted that all signals were made to have the same length N. In presence of missing data, at any point of the recording, missing samples were replaced by *NaN* values.

The filtered input is resampled to the sampling frequency f_s before applying:

$$y_0(n) = y_{\mathrm{F3}}(n) + y_{\mathrm{P3}}(n), \tag{1}$$

$$y_1(n) = y_{\mathrm{F4}}(n) + y_{\mathrm{P4}}(n), \tag{2}$$

where $y_{\mathrm{F3}}(n)$ denotes the filtered and resampled input from electrode $F3$, and $y_{\mathrm{P3}}(n)$, $y_{\mathrm{F4}}(n)$ and $y_{\mathrm{P4}}(n)$ are defined similarly.

The signal $y_m(n)$ is divided into L frames of length W overlapping by O samples, i.e, with an hop size of $R = W - O$. In each frame, and for each channel, F low-level features are computed. The list of features and their computation is summarised in Table 1, where $e(k, \ell)$ denotes the k-th feature extracted from the ℓ-th frame while $\mathbb{P}(\cdot)$ denotes the probability operator and $\mathrm{Q}_{0.75}(\cdot)$ the 75-th percentile. In each frame, peaks are detected and the standard deviation of the interval between peaks, the root mean square (RMS) of the difference between successive peaks and the number of peaks separated by less than 50 ms are added to the the features. These features are concatenated into a feature vector $\mathbf{e}_\ell$ of length $M \cdot F$, for each frame. The ordered sequence of these vectors is used as input to the predicting function that relies on a RNN with LSTM cells.

$e(\ell,\ 0+mF) = \min_{0 \leq n < W}\left(y_m(\ell R+n)\right)$
$e(\ell,\ 1+mF) = \max_{0 \leq n < W}\left(y_m(\ell R+n)\right)$
$e(\ell,\ 2+mF) = \frac{1}{W}\sum_{n=0}^{W-1} y_m(\ell R+n)$
$e(\ell,\ 3+mF) = \mathrm{median}_{0 \leq n < W}\left(y_m(\ell R+n)\right)$
$e(\ell,\ 4+mF) = \sqrt{\frac{1}{W}\sum_{n=0}^{W-1}(y_m(\ell R+n) - e(\ell,\ 2+mF))^2}$
$e(\ell,\ 5+mF) = e(\ell,\ 4+mF)^2$
$e(\ell,\ 6+mF) = \mathrm{E}\left\{(y - e(\ell,\ 4+mF))^4\right\}$
$e(\ell,\ 7+mF) = \mathrm{E}\left\{(y - e(\ell,\ 4+mF))^5\right\}$
$e(\ell,\ 8+mF) = \frac{\mathrm{E}\left\{(y-e(\ell,\ 4+mF))^3\right\}}{e(\ell,\ 4+mF)^3}$
$e(\ell,\ 9+mF) = \frac{\mathrm{E}\left\{(y-e(\ell,\ 4+mF))^4\right\}}{e(\ell,\ 4+mF)^4}$
$e(\ell,10+mF) = \sqrt{\frac{1}{W}\sum_{n=0}^{W-1} y_m(\ell R+n)^2}$
$e(\ell,11+mF) = \mathrm{Q}_{0.75}(\mathrm{y}) - \mathrm{Q}_{0.25}(\mathrm{y})$
$e(\ell,12+mF) = \sum_{n=0}^{W-1} y_m(\ell R+n)$
$e(\ell,13+mF) = e(\ell,\ 1+mF) - e(\ell,\ 0+mF)$
$e(\ell,14+mF) = -\sum_{n=0}^{W-1} \mathbb{P}(y_m(\ell R+n)) \log_2 \mathbb{P}(y_m(\ell R+n))$
$e(\ell,15+mF) = \frac{1}{L}\sum_{k=0}^{L-1} \lvert Y_m(k,\ell)\rvert^2$
$e(\ell,16+mF) = \frac{1}{L}\sum_{k=0}^{L-1} \lvert Y_m(k,\ell)\rvert$
$e(\ell,17+mF) = \mathrm{median}_{0 \leq k < L}\left(\lvert \mathrm{Y_m(k,\ell)}\rvert\right)$

Table 1: Summary of the low-level features extracted from each frame, not including features based on peak detection.

2.2. Predicting function

The previously described feature extraction results in a sequence of L vectors. Due to the padding of signals with *NaN* values, some of these vectors do contains invalid values. In order to ignore them, the proposed method first process the whole sequence through a so-called masking layer. Vectors containing invalid values will be ignored during both training and prediction, only the $\tilde{L}$ valid vectors are used for each signal. Hence, the predicting function uses the sequence of $\tilde{L} \leq L$ vectors $\mathbf{e}_\ell$ as input to predict a vector $\mathbf{p}$ of probabilities,

$$\mathbf{p} = [p_1,\ p_2,\ p_3,\ p_4,\ p_5],\ \text{with} \sum_{\mathrm{c}=1}^{5} \mathrm{p_c} = 1, \tag{3}$$

where $0 \leq p_c \leq 1$ denotes the estimated probability that the target patient was assigned a CPC score equals to c. For this purpose, the predicting function is implemented as a two layers RNN composed of LSTM cells, with a final output layer implemented as a dense layer with a softmax activation function. The roles of these layers is more formally described in the remainder of this subsection.

RNNs is a common extension of the conventional feed-forward artificial neural network (ANN). In a conventional feed-forward ANN, each λ-th layer applies a non linear mapping between input and output vectors, whereas the input of the λ-th RNN layer is an ordered sequence $\mathbf{X}^\lambda$ of T^λ input vectors $\mathbf{x}_\mathbf{t}^\lambda$, where $t \in [0, T^\lambda - 1]$ denotes the sequence index, i.e.,

$$\mathbf{X}^\lambda = \left\{\mathbf{x}_\mathbf{0}^\lambda, \mathbf{x}_\mathbf{1}^\lambda, \cdots, \mathbf{x}_{\mathbf{T}^\lambda\mathbf{-1}}^\lambda\right\}. \tag{4}$$

Each layer of an RNN computes a sequence $\mathbf{H}^\lambda$ of hidden vectors $\mathbf{h}_\mathbf{t}^\lambda$ of length $L_\mathbf{h}^\lambda$ and a sequence $\mathbf{Z}^\lambda$ of output vectors $\mathbf{z}_t^\lambda$ of length $L_\mathbf{z}^\lambda$, both containing T^λ vectors and defined similarly as in (4). The vectors in these sequences are computed by iteratively applying

$$\mathbf{h}_\mathbf{t}^\lambda = \mathcal{F}\left(\mathbf{W}_{\mathbf{x},\mathbf{h}}^\lambda \mathbf{x}_\mathbf{t}^\lambda + \mathbf{W}_{\mathbf{h},\mathbf{h}}^\lambda \mathbf{h}_{\mathbf{t-1}}^\lambda + \mathbf{b}_\mathbf{h}^\lambda\right), \tag{5}$$

$$\mathbf{z}_t^\lambda = \mathcal{F}\left(\mathbf{W}_{\mathbf{h},\mathbf{z}}^\lambda \mathbf{h}_\mathbf{t}^\lambda + \mathbf{b}_\mathbf{z}^\lambda\right), \tag{6}$$

where $\mathbf{W}_{\mathbf{x},\mathbf{h}}^\lambda$, $\mathbf{W}_{\mathbf{h},\mathbf{h}}^\lambda$ and $\mathbf{W}_{\mathbf{h},\mathbf{z}}^\lambda$ denote weight matrices of size $L_\mathbf{h}^\lambda \times L_\mathbf{x}^\lambda$, $L_\mathbf{h}^\lambda \times L_\mathbf{h}^\lambda$ and $L_\mathbf{z}^\lambda \times L_\mathbf{h}^\lambda$, respectively, and where $\mathbf{b}_\mathbf{h}^\lambda$ and $\mathbf{b}_\mathbf{z}^\lambda$ are bias vectors of length $L_\mathbf{h}^\lambda$ and $L_\mathbf{z}^\lambda$, respectively.

Applied to the prediction of the probability vector described in (3) from EEG signals, RNN have several advantages. On top of being able to take into account the temporal dependencies of the input, RNN can be applied to sequences of arbitrary length and therefore output prediction at different moments in the patient care. Additionally, in combination with the use of a masking layer, they are still able to output a prediction in presence of missing data. However, the values of the weight matrices and of the bias vectors still have to be learned during a training phase and the formulation in (5) and (6) can cause instability during training, leading to overly long training time or even divergence [8]. In order to avoid these issues, the so-called gated units of LSTM layers are used in our approach.

Though in a standard RNN layer, the function $\mathcal{F}(\cdot)$ in (5) is commonly a simple non-linear function such as a sigmoid, in an LSTM layer, this function relies on iterative updates of sequences of vectors, $\mathbf{I}^\lambda$, $\mathbf{O}^\lambda$, $\mathbf{F}^\lambda$ and $\mathbf{C}^\lambda$, the so-called, input gate, output gate, forget gate and cell memory. For each step t of the input sequence $\mathbf{X}^\lambda$, the vectors $\mathbf{i}_\mathbf{t}^\lambda$ and $\mathbf{f}_\mathbf{t}^\lambda$ are computed from the input vector $\mathbf{x}_\mathbf{t}^\lambda$ and from the memory cell vector $\mathbf{c}_\mathbf{t-1}^\lambda$ saved at the previous step, i.e.,

$$\mathbf{i}_\mathbf{t}^\lambda = \mathcal{S}\left(\mathbf{W}_\mathbf{x,i}^\lambda \mathbf{x}_\mathbf{t}^\lambda + \mathbf{W}_\mathbf{h,i}^\lambda \mathbf{h}_\mathbf{t-1}^\lambda + \mathbf{W}_\mathbf{c,i}^\lambda \mathbf{c}_\mathbf{t-1}^\lambda + \mathbf{b}_\mathbf{i}^\lambda\right), \quad (7)$$

$$\mathbf{f}_\mathbf{t}^\lambda = \mathcal{S}\left(\mathbf{W}_\mathbf{x,f}^\lambda \mathbf{x}_\mathbf{t}^\lambda + \mathbf{W}_\mathbf{h,f}^\lambda \mathbf{h}_\mathbf{t-1}^\lambda + \mathbf{W}_\mathbf{c,f}^\lambda \mathbf{c}_\mathbf{t-1}^\lambda + \mathbf{b}_\mathbf{f}^\lambda\right), \quad (8)$$

where $\mathcal{S}(\cdot)$ denotes the logistic sigmoid function. The resulting vectors $\mathbf{i}_\mathbf{t}^\lambda$ and $\mathbf{f}_\mathbf{t}^\lambda$ weight the influence of the current and previous input, respectively, to the updated vector $\mathbf{c}_\mathbf{t}^\lambda$ computed as

$$\mathbf{c}_\mathbf{t}^\lambda = \mathbf{f}_\mathbf{t}^\lambda \mathbf{c}_\mathbf{t-1}^\lambda + \mathbf{i}_\mathbf{t}^\lambda \tanh\left(\mathbf{W}_\mathbf{x,c}^\lambda \mathbf{x}_\mathbf{t}^\lambda + \mathbf{W}_\mathbf{h,c}^\lambda \mathbf{h}_\mathbf{t-1}^\lambda + \mathbf{b}_\mathbf{c}^\lambda\right). \quad (9)$$

The influence of this memory cell vector $\mathbf{c}_\mathbf{t}^\lambda$ to the layer output is weighted by the output gate $\mathbf{o}_\mathbf{t}^\lambda$ computed as

$$\mathbf{o}_\mathbf{t}^\lambda = \mathcal{S}\left(\mathbf{W}_\mathbf{x,o}^\lambda \mathbf{x}_\mathbf{t}^\lambda + \mathbf{W}_\mathbf{h,o}^\lambda \mathbf{h}_\mathbf{t-1}^\lambda + \mathbf{W}_\mathbf{c,o}^\lambda \mathbf{c}_\mathbf{t}^\lambda + \mathbf{b}_\mathbf{o}^\lambda\right), \quad (10)$$

and used to compute the hidden vector $\mathbf{h}_\mathbf{t}^\lambda$,

$$\mathbf{h}_\mathbf{t}^\lambda = \mathbf{o}_\mathbf{t}^\lambda \tanh\left(\mathbf{c}_\mathbf{t}^\lambda\right), \quad (11)$$

from which the output vector $\mathbf{z}_t^\lambda$ is finally computed as per (6).

In this application, the prediction function uses two LSTM layers, hence, the input to the first layer is,

$$\mathbf{X}^0 = \left\{\mathbf{e}_0, \mathbf{e}_1, \cdots \mathbf{e}_{\tilde{L}-1}\right\}, \quad (12)$$

and updates defined by equations (7) to (11) are applied to compute,

$$\mathbf{X}^1 = \left\{\mathbf{z}_0^0, \mathbf{z}_1^0, \cdots \mathbf{z}_{\tilde{L}-1}^0 \cdot\right\} \quad (13)$$

and the same updates are applied this sequence. The last vector $\mathbf{z}_{\tilde{L}-1}^1$ of the resulting sequence is used as input to the third and last layer, defined as a dense layer with a softmax activation function, i.e.,

$$\mathbf{p} = \mathbf{z}^2 = \mathcal{F}\left(\mathbf{W}_\mathbf{x,z}^\mathbf{2} \mathbf{z}_\mathbf{L-1}^\mathbf{1} + \mathbf{b}_\mathbf{z}^\mathbf{2}\right), \quad (14)$$

where $\mathcal{F}(\cdot)$ denotes the softmax activation function. Finally, the predicted CPC score is computed from $\mathbf{p}$ as,

$$\text{CPC score} = \sum_{c=1}^{5} c \cdot p_c, \quad (15)$$

and the probability of a poor outcome as $\sum_{c=3}^{5} p_c$. The expected outcome is labeled as *Poor* if this probability is higher than 0.5, and as *Good*, otherwise.

3. Experiments

3.1. Setup and parameters

The proposed method is implemented using Keras and all patients included in the training data were used. When reporting results (see next subsection) on the validation and test sets, for which training and evaluation were done by the challenge team, all of these patients were used for training and testing was done on data only available to the organizers. When reporting results on the training data, 5-fold cross-validation was used. In this case, patients in the training sets were separated into 5 non overlapping groups. Each group was alternatively used for testing while the others were used for training. Results are computed from the prediction obtained on all folds.

The parameters were set as fs =128 Hz while b_low, b_low, and b_util are set to 0.1, 30 and 50 Hz, respectively. Signals had a length $N = 72{\cdot}3600{\cdot}\ fs$, and frames of length $W = 60 \cdot f_s$ were used with 50% of overlap. Both LSTM layers had the same size, $L_\mathbf{h}^0 = L_\mathbf{h}^1 = 128$. In all cases, training was done using the Adam optimizer [9] for 50 epochs with a batch size of 4 (patients), aiming to minimize the cost function, set as the categorical cross-entropy between the predicted probability and the CPC scores. Prior to training, 10 % of the patients in the considered training data were set apart and used for validation. Weights were updated using the remaining 90 % of the data and the loss function on the validation data was computed after each epoch. Aiming to limit possible overfitting, dropout layers are used during training before each LSTM layer [10]. These layers are set so that only 50 % of the weights, randomly chosen, are updated during each epoch. For testing, we used the model obtained after the epoch that resulted in the lowest cost function over the validation part of the training data.

Training	Validation	Test	Ranking
0	0.045	0.025	36/36

Table 2: True positive rate at a false positive rate of 0.05 (the official Challenge score) for our final selected entry (team Oldenburg), including the ranking of our team on the hidden test set.

3.2. Results

The performance in terms of Challenge Score, i.e., true positive rate at a false positive rate of 0.05, is presented in Table 2. It appears that the proposed method only achieved a score of 25 on the hidden test set, ranking last out of the 36 submissions that managed to submit a successful entry. The performance in terms of the other metrics available in the evaluation tools provided by the challenge organizers, computed on the training set, are presented in Table 3.

4. Discussion and Conclusions

The proposed method uses time-varying features as input to a small-sized RNN based on LSTM cells. The features are computed using only 4 of the available EEG electrodes. The combination of a small amount of electrodes with a small sized model could make the method beneficial for a wide range of applications. Unfortunately, the performance measured in the scope of the Challenge appear disappointing. Tuning of parameters and better channel selection might improve the performance but some weaknesses inherent to the method can anyway be noted.

The features are extracted from the averaged input of electrodes. Even with a more beneficial electrode selection, this would still result in the spatial information being lost before feature extraction. Real data often contains missing data and the proposed method can still provide a prediction if this is the case. However, the chosen combination of signal padding and masking layer ignores the time at which the data is missing. As signals recorded at the time of prediction can be expected to be more meaningful than those recorded days prior, this may have a detrimental impact on the prediction.

Further analysis and comparisons with the other contributions to the Challenge will be great assets to improve the current performance.

Acknowledgments

This work was supported by the project Augmented Auditory Intelligence (A2I) funded by the German Ministry of Education and Research (BMBF) under grant number 16SV8594.

Challenge Score	0.05
Outcome AUROC	0.30
Outcome AUPRC	0.52
Outcome Accuracy	0.67
Outcome F-measure	0.63
CPC MSE	3.15
CPC MAE	1.48

Table 3: Results obtained using 5-fold cross-validation on the training set, using all metrics available in the evaluation tools provided by the challenge organizers.

References

[1] Goldberger AL, Amaral LA, Glass L, Hausdorff JM, Ivanov PC, Mark RG, et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation 2000;101(23):e215–e220.

[2] Reyna MA, Amorim E, Sameni R, Weigle J, Elola A, Bahrami Rad A, et al. Predicting neurological recovery from coma after cardiac arrest: The George B. Moody PhysioNet Challenge 2023. Computing in Cardiology 2023; 50:1–4.

[3] Amorim E, Zheng WL, Ghassemi MM, Aghaeeaval M, Kahndare P, Karukonda V, et al. The International Cardiac Arrest Research (I-CARE) Consortium Electroencephalography Database. Critical Care Medicine 2023; (in press).

[4] Bianco S, Napoletano P, Schettini R. Multimodal car driver stress recognition. In Proc. Int. Conf. on Pervasive Computing Technologies for Healthcare. Trento, Italy, 2019; 302–307.

[5] Bianco S, Napoletano P. Biometric recognition using multimodal physiological signals. IEEE Access 2019;7:83581–83588.

[6] Santos J, Falk T. Blind room acoustics characterization using recurrent neural networks and modulation spectrum dynamics. In Proc. Conf. Audio Eng. Soc. (AES). February 2016; .

[7] Cauchi B, Siedenburg K, Santos J, Falk T, Doclo S, Goetze S. Non-intrusive speech quality prediction using modulation energies and LSTM-network. IEEE Trans Audio Speech Lang Process July 2019;27(7):1151–1163.

[8] Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks. In Proc. Intl. Conf. Machine Learning (ICML). 2013; 1310–1318.

[9] Kingma D, Ba J. Adam: A method for stochastic optimization. In Proc. Int. Conf. on Learning Representations (ICLR). San Diego, CA, USA, 2015; .

[10] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 2014;15:1929–1958.

Address for correspondence:
Dr.-Ing. Benjamin Cauchi
OFFIS e.V., Escherweg 2, Oldenburg, Germany
benjamin.cauchi@offis.de