Python Implementation for Brain-Computer Interface Research by Acquiring and Processing the NeuroSky EEG Data for Classifying Multiple Voluntary Eye-Blinks

Oana-Andreea Rușanu(✉)

Product Design, Mechatronics and Environment Department, Transilvania University of Brasov, Brașov, Romania
oana.rusanu@unitbv.ro

Abstract. The Brain-Computer Interface (BCI) is a challenging research field reporting outstanding breakthroughs in biomedical engineering. This paper proposes a new BCI research-related solution by implementing customized Python scripts based on an artificial neural networks model to classify the raw electroencephalographic (EEG) signal detected by the embedded biosensor of NeuroSky portable headset. Achieving this aim is possible by applying features extraction techniques on the raw EEG data to generate the training dataset composed of 3000 recordings corresponding to executing simple, double, or triple voluntary eye-blinks. Detection of their specific EEG patterns resulted in calculating the following seven statistical features: mean, median, standard deviation, route mean square, the sum of values, Kurtosis Coefficient, and skewness. The voluntary eye-blinking proved to be the most precise and easily detected control signal in a BCI application to assist people with neuromotor disabilities. The proposed Python implementation of BCI software is practical, especially for the initial stages of research, by leveraging simple to use, inexpensive, and efficient instruments.

Keywords: BCI · Eye-blink · EEG · Python · ANN

1 Introduction

The Brain-Computer Interface (BCI) research field has considerably evolved during the last decades by proving efficient means of controlling the assistive devices, communication, sleep, and stress monitoring, especially for people with neuromotor disabilities. Their suffering is related to paralysis, spinal cord injuries, cerebral stroke, amyotrophic lateral sclerosis, or locked-in syndrome. These disorders determined the interruption of the neuronal pathway connecting the brain motor areas with the peripheral nerves and muscles. BCI is a promising solution for replacing the natural cerebral path with an artificial route allowing the translation of the thought into commands transmitted to an assistive robotic device. This desire is currently unachievable outside an experimental laboratory due to significantly higher requirements, including the complexity of the real-time neuronal biopotentials processing, the ability of the neuromotor disabled person to

I. Tiginyanu et al. (Eds.): ICNBME 2021, IFMBE Proceedings 87, pp. 666–672, 2022.
https://doi.org/10.1007/978-3-030-92328-0_84

perform BCI related cognitive tasks (motor imagery, concentration, mental calculus) to elicit recognizable patterns and the artificial intelligence techniques necessary to convert the intentions into action.

Therefore, BCI prototypes enabling affordable software and hardware solutions to improve the research results are still welcome and expected to overcome the existent issues by providing maximum accuracy, quick response, and rapid information transfer rate.

According to scientific literature [1, 2], implementing the most straightforward BCI system involves controlling an external device by detecting the voluntary eye-blinking across the electroencephalographic (EEG) signal. Moreover, the easiest way to acquire the raw EEG signal for developing versatile BCI applications is to use a portable headset such as NeuroSky Mindwave Mobile. The majority of the previous scientific articles [3] focused on calling on-the-shelf NeuroSky libraries providing convenient programming methods for achieving simple BCI applications with the expense of technical limitations for accomplishing advanced BCI research instruments. Therefore, several thresholding-based algorithms determined the measurement of the attention and meditation level and the eye-blink strength used as commands in a BCI for controlling: a wheelchair, a mobile robot, a robotic arm, a robotic hand, home appliances, and virtual simulations. Few papers [4] explored the NeuroSky EEG data acquisition and processing by applying different methods based on statistics, wavelet transform, supported vector machines, or artificial neural networks. In addition, the investigation of the mentioned methods targeted a specific BCI application not extendable to a general framework for conducting BCI research by enabling the fundamental phases: EEG data acquisition, processing, and classification.

Thus, it results in the main contribution of this paper by providing simple BCI research automated solutions implemented in Python open-source programming language to enable the NeuroSky EEG data processing, features extraction, training dataset generation, and artificial neural networks (ANN) based classification. Performing these stages resulted in performance assessment of the ANN accuracy for simple, double, and triple voluntary eye-blinks detection. Also, it resulted in an EEG dataset comprising 3000 sequences distributed as follows: 1000 sequences – one eye-blink detected, 1000 sequences – two eye-blinks detected, and 1000 sequences – three eye-blinks detected. Each sequence corresponds to the acquisition of 1024 raw EEG samples and the following seven extracted statistical features: mean, median, standard deviation, the sum of values, skewness, Kurtosis Coefficient, and route mean square (RMS). Unfortunately, the scientific literature reported very few EEG datasets [4] to test the algorithms of voluntary eye-blinking detection. The novelty of this paper resulted from customized Python scripts for enabling the acquisition, analysis, and classification of the raw EEG signal detected by the NeuroSky embedded biosensor. Further, the structure of the paper is: Section II presents an overview of the software and hardware systems used to implement a new BCI research solution, Section III explains the working principle and code instructions underlying the proposed instrument, Section IV discusses the achieved results and Section V summaries and concludes the paper.

2 Hardware and Software

2.1 Hardware System – NeuroSky Mindwave Mobile Headset

The BCI application developed in the current research involved raw EEG signal acquisition from the biosensor of the NeuroSky portable headset. The NeuroSky headset has an embedded sensor located to the prefrontal lobe, on the forehead, at the FP1 position according to the International 10–20 EEG System. Embedding the advanced ThinkGear based, the NeuroSky facilitates the initial stages of BCI research based on EEG data acquisition with a sampling rate of 512. Among the advantages provided by NeuroSky are the simple set-up, the ergonomic design, convenient working principle, and especially, free of charge raw EEG signal acquisition and recording to.csv files for further analysis.

2.2 Software System – Python Implementation Based on Raw EEG Data Acquisition, Processing, and ANN Classification

The current research contributes to the BCI field by implementing customized Python scripts to achieve the following phases (Fig. 1): cerebral biopotentials acquisition, raw EEG graphical displaying, statistical features extraction, EEG dataset generation, and classification of voluntary eye-blinks. There were necessary specific functions included by the following Python libraries: Numpy – to store data in arrays, Matplotlib – to create visualizations, SciPy – to calculate the statistical features, and Winsound – to get a sound.

3 Python-Based Customized Scripts for Enabling Multiple Voluntary Eye-Blinks Classification

3.1 Raw EEG Signal Acquisition and EEG Training Dataset Generation

NeuroPy library [5] was updated to run in Python 3. The rawValue variable stored each value of the raw EEG signal detected by the embedded NeuroSky biosensor. As a drawback noticed to the NeuroPy library, it is not easy to get precision regarding waiting time set to 2 ms between the raw EEG data samples. A healthy subject (girl, 29 years old) participated in the experiment involving 75 sessions of performing one, two, or three voluntary eye-blinks. Each session included 40 sequences representing 40 simple, double or triple voluntary eye-blinks. A recurrent time interval set to 2 s, introduced by a beep sound, determined a sequence of 1024 raw EEG samples.

It resulted in a.csv file comprising 40 sequences with 1024 samples, each organized in 40 rows and 1024 columns representing the raw EEG data. Therefore, the previously mentioned 75 sessions led to obtaining 75.csv files, including the $40 \times 1024 = 40960$ raw samples for each class: one-eye blink, two eye-blinks, and three eye-blinks. Thus, there were $3 \times 25 = 75$.csv files of raw EEG data and 75 graphical representations of the numerical content (40960 EEG raw samples).

It followed the extraction of seven statistical features from the raw EEG data containing 1024 samples and the generation of the EEG dataset necessary to classify the multiple voluntary eye-blinks. The extracted statistical features were: mean, median,

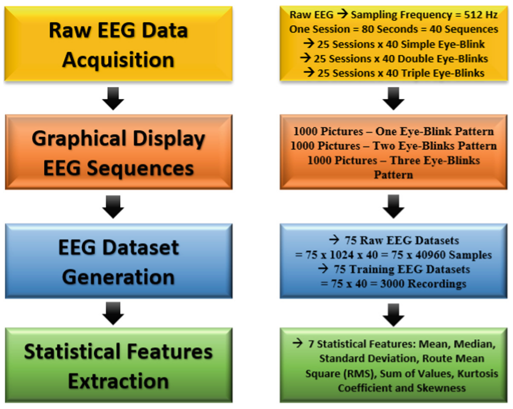

Raw EEG Data Acquisition
Raw EEG → Sampling Frequency = 512 Hz
One Session = 80 Seconds = 40 Sequences
→ 25 Sessions x 40 Simple Eye-Blink
→ 25 Sessions x 40 Double Eye-Blinks
→ 25 Sessions x 40 Triple Eye-Blinks
Graphical Display EEG Sequences
1000 Pictures – One Eye-Blink Pattern
1000 Pictures – Two Eye-Blinks Pattern
1000 Pictures – Three Eye-Blinks Pattern
EEG Dataset Generation
→ 75 Raw EEG Datasets
= 75 x 1024 x 40 = 75 x 40960 Samples
→ 75 Training EEG Datasets
= 75 x 40 = 3000 Recordings
Statistical Features Extraction
→ 7 Statistical Features: Mean, Median, Standard Deviation, Route Mean Square (RMS), Sum of Values, Kurtosis Coefficient and Skewness

Fig. 1. Python implementation of the main BCI related research stages

standard deviation, route mean square (RMS), the sum of values, skewness, and Kurtosis Coefficient. These seven selected features had to discriminate the executed eye-blinks initially set by the column representing the desired class: one eye-blink detected, two eye-blinks observed, and three eye-blinks recognized.

Thus, a single EEG dataset consisted of 8 columns (7 – features and 1 – class) and 40 rows (40 sequences of 2 s each for recording 1024 EEG samples). 75.csv files represented the 75 EEG datasets to classify the multiple voluntary eye-blinks (Fig. 2): 25 – simple eye-blinks, 25 – double eye-blink, and 25 – triple eye-blink.

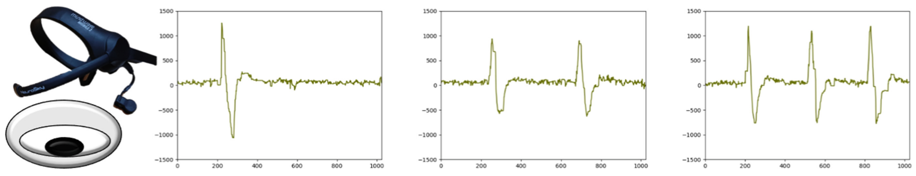


Fig. 2. Graphical displaying a sample from each class: simple (one), double (two) and triple (three) voluntary eye-blinks detected by NeuroSky Sensor

Concatenating all the 75 EEG datasets resulted in a single EEG dataset comprising $75 \times 40 = 3000$ sequences of eye-blinks. Then, the resulted.csv file consisted of 3000 rows and eight columns necessary to train an artificial neural network to classify the multiple voluntary eye-blinks.

3.2 Artificial Neural Networks Based Classification of Multiple Voluntary Eye-Blinks

Another contribution of the current research is implementing a customized Python script to employ the artificial neural network (Fig. 3) for multiple voluntary eye-blinks classification. The following Python libraries included the necessary specific functions: Numpy, Matplotlib, Pandas – to handle data analysis, Keras – to use the Tensorflow deep learning framework, and Scikit– for enabling Machine learning functions. After importing the essential libraries, it followed the uploading of the EEG dataset, including all the 3000 sequences and the seven calculated statistical features correlated with executing simple, double, and triple voluntary eye-blinks. Further, it followed the EEG dataset splitting into a training subset and a testing subset. Thus, the testing subset constituted 20% of the entire EEG dataset, resulting in 600 out of the 3000 samples only for testing purposes.

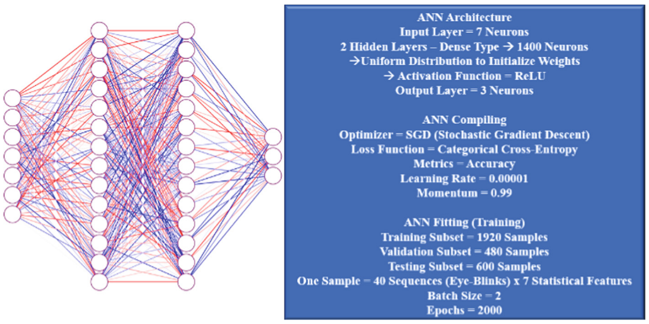
An essential stage consisted in creating the artificial neural network composed of four layers. The structure of the first and second hidden layers was the following: number of neurons = 1400, the uniform distribution used to initialize the weights, and activation function = ReLU. The input layer had seven neurons corresponding to the previously mentioned seven statistical features. The output layer had three neurons corresponding to the three classes regarding recognizing simple, double, and triple multiple voluntary eye-blinks.

Compiling the artificial neural network involved setting the following parameters optimizer = SGD (Stochastic Gradient Descent), loss (function) = categorical cross-entropy, and metrics = accuracy. The SGD convex function is used as Optimizer to determine the suitable set of weights by identifying a local minimum of the input function. Thus, it is necessary to set the learning rate to an appropriate value: LR = 0.00001. A momentum = 0.99 was also set to increase the speed of the optimization process.

The following critical stage consisted in fitting the compiled ANN to the generated EEG dataset. The ANN was trained on 1920 samples and validated on 480 samples. A sample is each of the 40 sequences/rows representing the measured values of the seven statistical features from each of the 75 EEG datasets. As mentioned previously, 20% representing 600 samples of the entire EEG dataset containing 3000 samples were necessary for testing purposes. It remained 2400 samples aimed for training and validation purposes. 20% of the 2400 samples or 480 EEG data sequences were necessary for validation purposes.

Then, considering the hyperparameters batch_size = 2 and epochs = 2000 and the rest of 1920 rows of training data, there were 960 batches with two samples each and 2000 passes through the whole EEG dataset. Each of the 2000 epochs or training iterations will involve 960 batches or 960 updates to the ANN-based model. After working through each batch of 2 samples, the internal ANN model parameters are updating. The learning algorithm works 2000 times through the entire EEG dataset.

Finally, the ANN model made predictions on the testing data composed of 600 samples. Also, there resulted in loss and accuracy specific to the testing process necessary to evaluate the performance of the ANN model. Also, the confusion matrix showed detailed results regarding the correctly and incorrectly detected samples from each of the three output classes: simple, double, and triple voluntary eye-blinks.



ANN Architecture
Input Layer = 7 Neurons
2 Hidden Layers – Dense Type → 1400 Neurons
→Uniform Distribution to Initialize Weights
→ Activation Function = ReLU
Output Layer = 3 Neurons

ANN Compiling
Optimizer = SGD (Stochastic Gradient Descent)
Loss Function = Categorical Cross-Entropy
Metrics = Accuracy
Learning Rate = 0.00001
Momentum = 0.99

ANN Fitting (Training)
Training Subset = 1920 Samples
Validation Subset = 480 Samples
Testing Subset = 600 Samples
One Sample = 40 Sequences (Eye-Blinks) x 7 Statistical Features
Batch Size = 2
Epochs = 2000

Fig. 3. Python artificial neural network for simple, double and triple voluntary Eye-Blinks classification

4 Results and Discussions

The proposed artificial neural networks-based architecture aimed for multiple voluntary eye-blinks classification reported the results in Table 1.

Table 1. Results for training, validating, and testing the Python ANN

EEG Dataset	Number of Eye-Blinks Sequences	Loss (%)	Accuracy (%)	Correctly Detected Eye-Blinks Sequences
Training	1920	0.56	100.00	1920
Validation	480	33.27	92.92	446
Testing	600	31.21	92.33	554

A video demonstration of the Python-based implementation for voluntary eye-blinks classification is available at the YouTube Unlisted Links: https://youtu.be/tvj8UcGbWa8 and https://youtu.be/MeMMj2JNUfk. Although the obtained results reported high values of accuracy for training, validation, and testing the compiled Python-based ANN, further experiments are necessary to analyze possible particularities implied by a large number of individuals taking into account their age, gender, stress level, degree of disability or the ability to focus on accomplishing the required task – to execute simple, double or triple eye-blinks. The customized ANN-based architecture aims to classify eye-blinks characterized by average amplitude. The ANN may need improvements to differentiate between voluntary eye-blinks of various strengths depending on the effort of the eyeball muscles.

Otherwise, the presented Python automated scripts aim to offer a general-purpose BCI research instrument as long as the conducted experiments involve the following three fundamental stages: raw EEG data acquisition, processing enable by features extraction, and ANN-based classification. Still, the proposed Python implementation aims to enable the ThinkGear embedded chip of the most affordable and portable EEG commercial headset – NeuroSky Mindwave Mobile. In addition, the presented Python software tool proves its usefulness for any application that involves the assessment of the seven statistical features: mean, median, standard deviation, route mean square, the sum of values, Kurtosis Coefficient, and skewness extracted from the raw EEG signal detected by the biosensor placed on the forehead.

5 Conclusions

This paper presented a Python-based implementation of a simple BCI-related research instrument necessary to acquire, process, and classify the raw EEG signal detected by the embedded sensor of the NeuroSky headset. A customized ANN-based architecture classified the multiple voluntary eye-blinks used as control signals in a straightforward brain-computer interface application. It resulted in the generation of a training dataset containing 3000 recordings evenly distributed for detecting simple, double, and triple voluntary eye-blinks. It also involved extracting the following statistical features from the raw EEG signal: mean, median, standard deviation, route mean square, the sum of values, Kurtosis Coefficient, and skewness. The proposed Python application provides simplicity and efficiency to help researchers explore and experiment with the working principle underlying the BCI scientific field.

Future research should update the ANN-based framework to classify voluntary eye-blinks of various strengths: mild, regular, or firm. Also, the Python-based ANN should differentiate between spontaneous, reflexive, and voluntary eye-blinks. Moreover, the BCI instrument requires improvements to detect winks precisely. Extracting additional statistical features will extend the universality of the experiments conducted with the proposed BCI software solution. The ultimate goal is to achieve a real-time running of the Python-based voluntary eye-blinks classification.

Conflict of Interest. The authors declare that they have no conflict of interest.

References

1. Rejer, I., Cieszyński, L.: RVEB—an algorithm for recognizing voluntary eye blinks based on the signal recorded from prefrontal EEG channels. Biomed. Signal Process. Control **59**, 101876 (2020). https://doi.org/10.1016/j.bspc.2020.101876, ISSN 1746-8094
2. Sharma, K., Jain, N., Pal, P.K.: Detection of eye closing/opening from EOG and its application in robotic arm control. Biocybern. Biomed. Eng. **40**(1), 173–186 (2020). https://doi.org/10.1016/j.bbe.2019.10.004, ISSN 0208-5216
3. Prem, S., Wilson, J., Varghese, S.M., Pradeep, M.: BCI integrated wheelchair controlled via eye blinks and brain waves. In: Pawar, P.M., Balasubramaniam, R., Ronge, B.P., Salunkhe, S.B., Vibhute, A.S., Melinamath, B. (eds.) Techno-Societal 2020, pp. 321–331. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-69921-5_32
4. Lo Giudice, M., et al.: 1D Convolutional Neural Network approach to classify voluntary eye blinks in EEG signals for BCI applications. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–7 (2020). https://doi.org/10.1109/IJCNN48605.2020.9207195
5. Neuropy Library. https://github.com/lihas/NeuroPy.git. Accessed 14 Aug 2021