Improving Speller BCI performance using a cluster-based under-sampling method

Sergio A. Cortez
Dept. of Electrical Engineering
Universidad de Ingeniería y Tecnología
Lima, Peru
sergio.cortez@utec.edu.pe

Christian Flores
Dept. of Electrical Engineering
Universidad de Ingeniería y Tecnología
Lima, Peru
cflores@utec.edu.pe

Javier Andreu-Perez
Dept. of Electronic Engineering
University of Essex
Essex, United Kingdom
javier.andreu@essex.ac.uk

***Abstract*—A Brain-Computer Interface (BCI) allows its user to interact with a computer or other machines by only using their brain activity. People with motor disabilities are potential users of this technology since it could allow them to interact with their surroundings without using their peripheral nerves, helping them regain their lost autonomy. The P300 Speller is one of the most popular BCI applications. Its performance depends on its classifier's capacity to identify and discriminate the presence of the P300 potentials from electroencephalographic (EEG) signals. For the classifier to do this correctly, it is necessary to train it with a balanced data-set. However, as the P300 is usually elicited with an oddball paradigm, only unbalanced distributions can be obtained. This paper applies an under-sampling method based on Self-Organizing Maps (SOMs) on P300 EEG signals looking to increase the classifier's accuracy. Two classifying models, a deep feedforward network (DFN) and a deep belief network (DBN), are tested with data-sets obtained from healthy subjects and post-stroke victims. We compared the results with our previous works and observed an increase of 7% in classification accuracy for our most critical subject. The DBN achieved a maximum classification accuracy of 95.53% and 94.93% for a healthy and post-stroke subject, while the DFN, 96.25% and 93.75%.**

***Index Terms*—brain-computer interface, neural networks, self-organizing maps, post-stroke, EEG**

I. Introduction

A brain-computer interface (BCI) is a technology that translate people's brain activity into commands to control machines [1]. The brain signals are recorded using different brain scan techniques, such as electroencephalography (EEG) and magnetic resonance imaging (MRI) and then translated using a classifier for an specific application. EEG is usually employed in many applications due to its non-invasive character and low cost compared to other techniques. Motor-impaired people are potential users of BCI technology since it can provide them with alternative means to interact with their surroundings, improving their lifestyle [2]. The *Speller* is a well-known BCI application first proposed by [3] that acts as an alternative communication tool using the P300 event-related potential (ERP). An oddball paradigm is commonly used to elicit the P300 waveform and it consists of presenting the user target stimuli blended among irrelevant stimuli. It takes the name P300 because it appears in the user's EEG signals as a positive deflection 300ms after the target stimulus is presented. The simplicity of the oddball paradigm makes it useful for developing quick solutions [4].

However, the main drawback of using this paradigm is that it provides unbalanced data-sets. Unbalanced distributions negatively affect the classifier's accuracy because the resulting model tends to classify only the majority class. There are several works focused on how to deal with this kind of data-sets and how to balance them. The work of [5] presented a method for balancing two-class data-sets using an under-sampling approach, that is, to discard samples from the majority class. It consisted of selecting a representative subset of the majority samples based on their distance to the minority class's boundaries. Testing it with a neural network and an oil-slick data-set, they managed to increase the classification accuracy by 4%. In [6], the authors presented several cluster-based under-sampling methods. Using the *k-means* algortihm, they selected a subset of both classes that best represented the data-set's structure. Their best approach increased the classifier's precision for the minority class by 15% when comparing it using the classical random under-sampling approach. The under-sampling method proposed by Vannucci [7] used Self-Organizing Maps (SOMs) to cluster both classes and remove the samples that lie close to the regions dominated by the minority class. Compared to the random under-sampling approach, their method increased a decision tree classifier's sensitivity by 24%, testing it with the UC Irvine's satellite data-set. Vannucci's approach has the characteristic it improves the classifier's sensitivity without compromising its specificity without changing the balance rate drastically.

Machine learning algorithms have been successfully used for developing classifiers that identify and discriminate user's commands automatically [8]. These classifiers are of particular importance since their performance determines the BCI's correct functioning. Classical machine learning algorithms, like Support Vector Machines (SVM), have the main disadvantage that they often require feature reduction pre-processing stages to achieve a good classification accuracy [9] [10]. These stages are usually computationally expensive and have an impact on the system's response time. Deep learning algorithms have the advantage of not requiring complex pre-processing stages because their learning scheme allows them to identify relevant features automatically from raw data. Regarding deep learning techniques applied to BCI applications, in [11] the authors used a convolutional neural network (CNN) for P300 classifi-

978-1-7281-2547-3/20/$31.00

cation. They were able to achieve an accuracy of 95.5% using target by block. In [12], the authors used a deep belief network (DBN) to also classify P300 trials and reported a classification accuracy 86.4%. In the work of [13], a deep feedforward network (DFN) was used to classify motor imagery data obtained with EEG and functional near-infrared spectroscopy (fNIRS). They reported a maximum classification accuracy of 94.6%. The architecture employed in the last two neural networks makes them having shorter training periods compared to CNN. This benefit makes them suitable for designing and testing different configurations faster without requiring high computational power.

In this paper, we applied the method proposed by Vannucci [7] to increase the accuracy of classifiers to be used in a P300-based BCI for stroke victims. Two deep neural network architectures are proposed for P300 single-trial classification and their performance is tested using data obtained from healthy subjects and post-stroke patients.

This work is organized as follows: in section II the materials and methods used for the training process of the classifiers are described. In section III the results are analyzed and compared with our previous works. Finally, we present the conclusions and future work in section IV.

II. Materials and Methods

A. Participants and EEG Acquisition

Nine subjects agreed to participate in this study. Table I shows the age, gender, and medical diagnosis of each participant. The healthy participants acted as the control group for the patient cohort. Subject S07 presented hemiplegia and severe aphasia. Subjects S08 and S09 exhibited mild aphasia, but only subject S09 showed moderate apraxia limited to his lower extremities. The ethics committee of the Universidad Peruana Cayetano Heredia issued the ethical approval for the experiment and informed written consent. All participants were informed about the academic objectives pursued in this study and ensured the preservation of their anonymity.

The EEG signals were recorded using sixteen bipolar electrodes and a g.USBamp amplifier (g.tec medical engineering GmbH, Austria) at 2400 Hz. The electrodes were placed following the 10-20 system on positions: Fz, FC1, FC2, C3, Cz, C4, CP1, CP2, P7, P3, Pz, P4, P8, O1, O2, and Oz. The ground electrode was placed at the subject's right mastoid and, the reference electrode, on its left earlobe. More information about the experimental setup can be reviewed in [14]. The Fig. 1 shows the timing scheme of the experiment and a brief description.

B. Signal Pre-processing

The EEG signals were downsampled from 2400 Hz to 120 HZ and any spurious spectral components were removed using notch filters. Then, signals passed through a sixth order Butterworth filter with cut-off frequencies in 1 and 15 Hz. The data points recorded in one second after an image was presented on the screen were selected and stored in a 16-by-120 array, defining a trial. According to the experimental setup, on average up to 3200 trials were recorded per subject. Finally, any outliers were eliminated by winsorization [4], and then signals were standardized.

TABLE I: Participants information

Subject	Age	Gender	Diagnosis
S01	33	Male	Healthy
S02	21	Male	Healthy
S03	20	Male	Healthy
S04	21	Male	Healthy
S05	24	Male	Healthy
S06	29	Male	Healthy
S07	20	Male	Hemorrhagic post-stroke
S08	52	Female	Ischemic post-stroke
S09	55	Male	Ischemic post-stroke

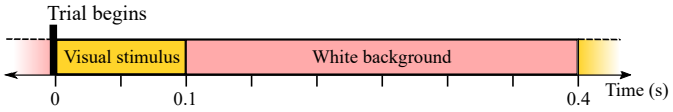

Fig. 1: Protocol's time scheme. One of six images is randomly selected and displayed on a screen for 100ms, followed by a white background for 300ms. After displaying all six images once, the process repeats itself randomly between 20 and 25 times.

C. Feature Vectors and Balancing Process

The feature vectors were constructed rearranging the data of the subject's trials. The channels were first demeaned and then concatenated forming a row vector. Each trial and thus, each feature vector, has a label which indicates if its related visual stimulus triggered a P300 response. If that is the case, the trials are called *target*, and if not, *non-target*. However, since consecutive trials overlap, the adjacent *non-target* trials to the *target* ones may also present the P300 waveform or a part of it. These type of trials were removed before applying the balancing process.

The ratio between the *target* and *non-target* trials is 1 to 5. This uneven distribution was balanced using Self-Organizing Maps (SOMs). Proposed by Kohonen [15], SOMs are unsupervised machine learning algorithms focused on clustering and pattern recognition tasks. These shallow neural networks map high dimensional inputs onto a discreet topological 2-D space for easier characteristics analysis and visualization. The neurons or clusters that make up the topological space compete against each other for possession of the inputs the SOMs receive and thus, strengthen its relationship with certain data characteristics [16]. The learning principle of SOMs is briefly presented below.

Let us define a set U whose members are the indexes of N output neurons arranged in an grid. Also, let $\boldsymbol{w}_i(t)$ be the weight vector associated to the ith output neuron (ranging from 1 to N) at a given time and $\boldsymbol{r}_i$ its position vector on the grid. The weight vectors have the same dimension as the inputs of the SOM and their values are initialized with random numbers, also in the same range of the inputs. The training process is divided into two parts, which are repeated a defined number

of times. First, a winning neuron is selected by comparing the euclidean distance between all weight vectors and a given certain observation $\boldsymbol{x}(t)$:

$$c(t) = \arg\min_{j \in U} \|\boldsymbol{w}_j(t) - \boldsymbol{x}(t)\|. \quad (1)$$

A weight vector $\boldsymbol{w}_c(t)$, associated to the winning neuron, solves Eq. 1. The second part of the training process updates all weight vectors by applying the following learning rule:

$$\boldsymbol{w}_j(t+1) = \boldsymbol{w}_j(t) + \alpha(t)\rho(c,j,t)[\boldsymbol{x}(t) - \boldsymbol{w}_c(t)], \quad (2)$$

where $\alpha(t)$ is the learning rate parameter and $\rho(c,j,t)$ is the neighborhood function which simulates the lateral interconnections between the winning neuron and all its neighbors. The learning rate is usually defined as a decreasing linear function whose values are greater than 0 and less than 1. The neighborhood function is the Gaussian $\rho(c,j,t) = \exp(-\frac{\|\boldsymbol{r}_c - \boldsymbol{r}_j\|^2}{2\sigma^2(t)})$, where σ is the neighborhood radius and depends on the grid where the neurons are arranged. The Eq. 2 essentially shift the weight vectors of the surrounding neurons towards the winning one, making them more likely to recognize similar characteristics and thus, fostering competition. After a determined amount of epochs, the distance between neurons in the grid (norm between weight vectors) and the number of times they won (hits) can be analyzed searching for patterns.

Following the under-sampling approach proposed by Vannucci [7], two SOMs were used to cluster both classes in the subject's data separately. The steps of the balancing process are presented in Algorithm 1. The main idea of this method is to independently generate regions predominantly occupied by each class in the same domain to identify and remove those *non-target* samples that lie close or are similar to the *target* ones. The centroids of each cluster, which are the weight vectors, are used to calculate a ranking for each *non-target* vector to determine the candidates for removal. The authors designed this method to increase the domain zones the classifier would associate to the *target* samples, resulting in fewer classification conflicts and improved classifier performance.

The number of neurons in each SOM an the value of β were empirically tuned. The SOM used to cluster the *target* samples had a 4-by-4 hexagonal grid and the one used for the *non-target* samples had a 6-by-6 hexagonal grid. β was set to 0.6 after testing different values evaluating the classifier's performance. The training stage of both SOMs was fixed for 200 epochs. Fig. 2 shows the main characteristics of both SOMs after training. The resulting vectors selected were stacked shaping the training and testing matrices for the classifiers.

D. Classification Models

We used two neural networks: a deep feed-forward network (DFN) and a deep belief network (DBN). Such networks work as function approximators which can be used for binary classification [17]. In training, their internal parameters are adjusted trying to minimize the error between actual outputs and target outputs using examples input-output pairs. Each

Algorithm 1 Balancing process [7]

1. Determine the number of non-target samples K to be maintained to achieve a balanced dataset.
2. Normalize the inputs from 0 to 1.
3. Define the number of neurons of the two SOMs and train them using the *non-target* and *target* samples respectively.
4. Extract the weights or centroids from each neuron in both SOMs and store them in W_r for the *target* samples and W_f, for the *non-target* ones.
5. For each non-target vector v_f, obtain its score defined as:

$$score(v_f) = \beta(\min_{\forall x \in W_r} \|v_f - x\| - \min_{\forall y \in W_f} \|v_f - y\|) + (1-\beta)(\operatorname{mean}_{\forall x \in W_r} \|v_f - x\| - \operatorname{mean}_{\forall y \in W_f} \|v_f - y\|),$$

   where the β value and its complement relate the distances from the *non-target* sample to each class's centroids and their average, respectively.
6. Rank all v_f samples according to their score and maintain those whose scores are higher than the rest.
7. Merge both classes and shuffle the dataset.

neuron learns relevant characteristics from the input data and strengthens their relationship with other neurons through their synaptic weights making the whole network able to classify new similar data. A DFN typically has multiple hidden layers and it is trained using back-propagation. Considering fully connected layers and differentiable transfer functions, its training consists on minimizing a performance function using a gradient method [18]. The network is updated with each training sample until optimal performance is achieved or a specific number of epochs has passed.

A DBN is made of two or more restricted Boltzmann machines (RBMs) joined together [19]. A RBM is neural network with an input (or visible) layer fully connected to a single hidden layer. The DBN training is divided in two stages: the unsupervised training (or pretraining) of each RBM and the supervised training of the whole network. The pretraining is motivated by some problems related to regular training by back-propagation. Specifically, the minimization of the performance function by a gradient method gets more complex when the networks have multiple layers due to its non-linear character. The unsupervised greedy layer-wise pretraining initializes the RBM's parameters to increase the probability of the gradient reaching a global maximum (or minimum) when the supervised training takes place. During this stage, each RBM learns to identify the most relevant characteristics from its inputs through the contrastive divergence (CD) algorithm [20]. The RBMs are trained in consecutive order, meaning when the first one finishes its training, its hidden states serve the next RBM as its inputs. After this stage, the whole network is fine-tuned using error back-propagation. The simplified unsupervised training algorithm is briefly presented below.

Let us assume a binary RBM, which then can be used to generalize a model for real value inputs [21]. Let $\boldsymbol{v}$ and $\boldsymbol{h}$

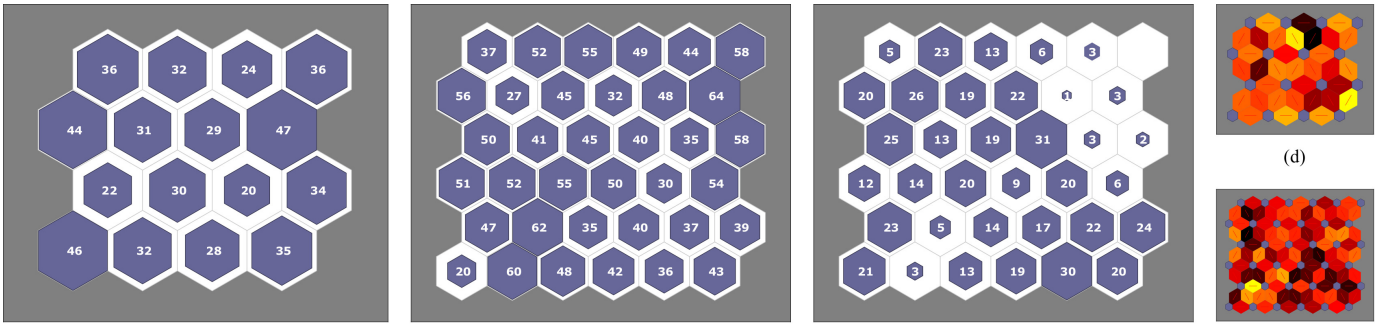

Fig. 2: *Target* and *non-target* self-organizing maps training results using subject S01 data. Fig. 2a shows the grid of the SOM trained with *target* trials and presents the number of hits of each neuron. Fig. 2b also shows the grid and the number of hits but with the SOM trained with *non-target* trials. Fig. 2c shows the selected *non-target* trials using Algorithm 1 in their respective neuron cluster. Figs. 2d and 2e show the distances between neurons (blue hexagons) of both SOMs from Figs. 2a and 2b respectively. A darker color indicates a larger distance while a lighter one, the opposite.

be the state vectors of the visible and hidden neurons of the RBM. Also, let $\boldsymbol{w}$ be the weight matrix which represents the interaction between the ith visible neuron and the jth hidden neuron whilst b_i and c_j are their bias terms. The energy of a joint configuration $(\boldsymbol{v}, \boldsymbol{h})$, is defined as:

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{V}\sum_{j=1}^{H} v_i h_j w_{ij} - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} c_j h_j,$$

where V and H are the total number of visible and hidden neurons respectively. The probability the neural net assigns to an input $\boldsymbol{v}$ is computed by summing all its hidden states, resulting in

$$p(\boldsymbol{v}) = \frac{1}{Z}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})},$$

where $Z = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$ is the partition function.

An RBM can be assigned to a specific input by modifying its weights and biases. The derivative of the log probability of an input vector with respect to the RBM's weights can be used to define the following learning rule:

$$\frac{\partial \log p(\boldsymbol{v})}{\partial w_{ij}} \propto \Delta w_{ij} = \eta(\langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{reconst}),$$

where η is the learning rate (or step) and the terms in the angle brackets are the expectations under the distributions of the training set and its reconstruction respectively. The $\langle v_i h_j\rangle_{data}$ term can be calculated, for a $\boldsymbol{v}$ input, using the conditional probability

$$p(h_j = 1|\boldsymbol{v}) = f(c_j + \sum_{i=1}^{V} v_i w_{ij}), \tag{3}$$

where $f(x) = \frac{1}{1+e^{-x}}$ (logistic sigmoid). The second term $\langle v_i h_j\rangle_{reconst}$ can be calculated using Eq. 3 with the reconstructions after these are computed applying

$$p(v_i = 1|\boldsymbol{h}) = f(b_i + \sum_{j=1}^{H} h_j w_{ij}) \tag{4}$$

to the hidden states obtained when calculating $\langle v_i h_j\rangle_{data}$. To summarize, the training goal is to reconstruct, as similar as possible, the inputs using only the hidden states by adjusting the network's parameters. Real input values are processed using a Gaussian–Bernoulli RBM, which has a variation in its energy function and the conditional distribution described in Eq. 4 is modeled with a Gaussian instead [22].

E. Classifier Architecture and Training

The architecture of the two networks is shown in Fig. 3. Both have four fully connected hidden layers of 60, 40, 30 and, 20 neurons respectively but different classification layers. For the DFN: each neuron's activation function was modeled using a hyperbolic tangent (tanh) sigmoid except for the single output neuron used for classification, which used a logistic sigmoid function. The feature vectors entering the network were re-normalized from -1 to 1. The optimizer selected for training was the scaled conjugate gradient method. The training was fixed for 600 epochs to ensure full convergence for all subjects and had an initial learning rate of 0.2. The training of each subject's DFN took approximately five minutes.

For the DBN: four RBMs were stacked, making up the input and hidden layers of the network. The first RBM was type Gaussian - Bernoulli, and the rest, Bernoulli - Bernoulli ones. The activation function of each neuron was modeled using a logistic sigmoid. The DBN's classification layer employed two neurons with a softmax activation function. The feature vectors entering the network were re-normalized from 0 to 1. The RBMs were trained using CD, the learning step was set to 0.01 with an initial momentum of 0.5 and a final moment of 0.9, following the work of [23]. The pretraining was fixed for 100 iterations and took approximately ten minutes per subject. The DBN supervised training process was similar to the DFN

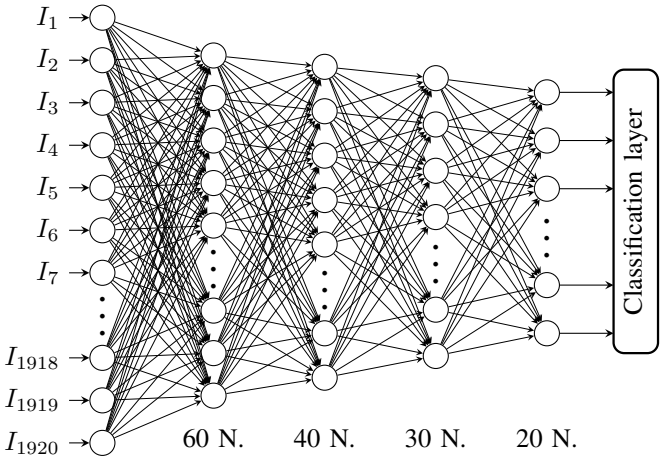

Fig. 3: Architecture of the neural networks proposed for P300 classification. The classification layer for the DFN is a single neuron and, for the DBN, are two.

one. A Nvidia GTX 1050 GPU and an Intel core i7 CPU were used for the computations on MATLAB R2019a.

III. Results and Discussion

Table II shows the cross-validated classification accuracy and its standard deviation using 5-folds for each model. Both networks achieved a classification accuracy higher than 90% for the subjects S01, S03, S08 and S09. The DBN obtained better results than the DFN for the post-stroke victims, whilst for the healthy ones, both networks presented similar performance. The DBN's pretraining stage was an essential factor in its supervised training since it compensated for the disadvantages of using logistic sigmoids as activation functions. This function is not usually employed to model neural networks because, in training, they provide weaker gradients than other functions such as the hyperbolic tangent, which can compromise the network's performance. [24]. The initialized weights and biases, obtained through the CD algorithm, allowed the DBN to converge during the supervised learning stage more efficiently than the DFN.

Concerning the patient cohort, subjects S09 and S08 were able to achieve an accuracy higher than most healthy subjects; however, subject S07 performance was the lowest among all. The results obtained with that subject may be related and caused by its critical medical condition. It is well-known that the P300 waveform is a visual-induced endogenous brain response that can be elicited without problems in post-stroke victims; nonetheless, if it is evoked or not depends entirely on the subject's concentration. The subject S07 probably has a short attention span consequence of the stroke, which made its concentration on the experiment diminish over time. Both subjects S08 and S09 results indicate a P300 based BCI for ischemic post-stroke victims employing these classifiers will work properly.

Table III shows the classification accuracy obtained in our previous works [25] [26] [14] for the same subjects using also

TABLE II: Cross-validated model accuracy

Subject	SOM under-sampling method	
	DBN	DFN
S01	95.53 (1.41)	**96.25 (1.58)**
S02	**84.83 (3.68)**	84.17 (1.56)
S03	**91.35 (1.66)**	91.17 (1.61)
S04	**87.38 (1.31)**	84.52 (2.91)
S05	87.37 (2.16)	**89.85 (1.86)**
S06	84.51 (1.47)	**88.07 (1.64)**
S07	**77.58 (1.49)**	73.66 (4.05)
S08	**93.66 (0.54)**	91.11 (1.30)
S09	**94.93 (1.02)**	93.75 (1.68)

a DBN and a DFN, besides a support vector machine (SVM) and an ANFIS classifier. These classifiers used a data-set balanced through random under-sampling and were validated using a 5-fold cross-validation method. The ANFIS classifier was not trained with subject S05 because, for that moment, the recording process of its EEG signals was still ongoing. Both deep networks trained with the data-set balanced using SOMs outperformed their counterparts and the other classifiers previously mentioned for all subjects, specifically in the patient cohort. Although the networks shared the same architecture and a similar supervised training stage, the accuracy was not even close to most of the subjects. These results prove that the trial selection criteria in the balancing process can positively or negatively impact the classifier's performance. The cluster-based balancing process using SOMs applied here aimed to remove the *non-target* trials that were alike or share similar characteristics to most of the *target* ones. Such trials can be generated by a momentary loss of attention due to external or internal events, although recorded under similar circumstances to the rest. Ideally, these kinds of samples should not represent the majority of its class, but since we are dealing with post-stroke victims, it may be the case and require an appropriate criterion for discarding them. A classifier can efficiently discriminate the class of samples in a data-set when there are significant differences between them. The significant improvement in classification accuracy for the stroke victims and most healthy subjects evidence the importance of the balancing process.

The two networks presented in this work will be used in BCIs, such as the ones proposed by [4] [11], to classify trials using a target by block approach instead of single-trials only. The oddball paradigm that is commonly employed requires a block classification to identify the user's commands for a specific application. The single-trial classification accuracy is of special concern since it can reduce the amount of time needed for the system to achieve a 100% target by block classification accuracy. Such characteristic is fundamental in a BCI since it could help its user in tasks that require fast decisions, like answering the phone.

TABLE III: Previous works results

Subject	Random under-sampling method			
	DBN [25]	DFN [26]	SVM [26]	ANFIS [14]
S01	91.6	91.8	91.5	85.3
S02	80.7	80.3	79.1	77.4
S03	85.8	85.3	83.9	79.3
S04	81.7	75.7	78.9	79.5
S05	83.5	84.9	83.4	-
S06	82.5	83.0	81.7	72.4
S07	66.6	68.6	69.2	70.1
S08	88.1	89.6	85.5	74.9
S09	85.8	86.9	87.4	78.4

IV. Conclusions

Two P300 single-trial classifiers were presented and tested on six healthy subjects and three post-stroke patients. Both classifiers can be used in the design of a P300-based BCI targeted at stroke victims. However, subjects with severe medical conditions may require variations in the paradigm used to record their EEG signals to prevent attention loss from diminishing their performance. Both networks tested have the advantage of not requiring computing-intensive pre-processing stages. Their deep architecture allowed them to learn and generalize efficiently, basically from raw data. The balancing process proposed by Vannucci [7] was proven to be effective with EEG signals. The appropriate selection of trials was critical for increasing the classification accuracy, specifically in the patient cohort. The obtained results outperformed our previous works, which used a random under-sampling approach for balancing. The classification accuracy with the most critical subject data was improved by 7%.

As an alternative approach to deal with critical patients, we recommend reducing the number of runs in each session, making the daily recording periods shorter at the expense of taking more days to obtain the necessary samples. As future work, we intend to test the classifiers using target by block and test a BCI online. We also intend to include and test amyotrophic lateral sclerosis (ALS) patients data with different deep learning architectures looking to increase the classification accuracy.

Acknowledgment

This work was supported by the Programa Nacional de Innovacion para la Competitividad y Productividad of Peru, under the grant PIAP-3-P-483-14 and a seed fund of the Universidad de Ingeniería y Tecnología.

References

[1] H.-J. Hwang, S. Kim, S. Choi, and C.-H. Im, "Eeg-based brain-computer interfaces: a thorough literature survey," *International Journal of Human-Computer Interaction*, vol. 29, no. 12, pp. 814–826, 2013.

[2] E. W. Sellers and E. Donchin, "A p300-based brain–computer interface: initial tests by als patients," *Clinical neurophysiology*, vol. 117, no. 3, pp. 538–548, 2006.

[3] L. A. Farwell and E. Donchin, "Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials," *Electroencephalography and clinical Neurophysiology*, vol. 70, no. 6, pp. 510–523, 1988.

[4] U. Hoffmann, J.-M. Vesin, T. Ebrahimi, and K. Diserens, "An efficient p300-based brain–computer interface for disabled subjects," *Journal of Neuroscience methods*, vol. 167, no. 1, pp. 115–125, 2008.

[5] M. Kubat, S. Matwin, *et al.*, "Addressing the curse of imbalanced training sets: one-sided selection," in *Icml*, vol. 97, pp. 179–186, Citeseer, 1997.

[6] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5718–5727, 2009.

[7] M. Vannucci and V. Colla, "Self–organizing–maps based undersampling for the classification of unbalanced datasets," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, IEEE, 2018.

[8] F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy, and F. Yger, "A review of classification algorithms for eeg-based brain–computer interfaces: a 10 year update," *Journal of neural engineering*, vol. 15, no. 3, p. 031005, 2018.

[9] S. Kundu and S. Ari, "P300 detection with brain–computer interface application using pca and ensemble of weighted svms," *IETE Journal of Research*, vol. 64, no. 3, pp. 406–414, 2018.

[10] X. Li, X. Chen, Y. Yan, W. Wei, and Z. J. Wang, "Classification of eeg signals using a multiple kernel learning support vector machine," *Sensors*, vol. 14, no. 7, pp. 12784–12802, 2014.

[11] H. Cecotti and A. Graser, "Convolutional neural networks for p300 detection with application to brain-computer interfaces," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 3, pp. 433–445, 2010.

[12] Z. Lu, N. Gao, Y. Liu, and Q. Li, "The detection of p300 potential based on deep belief network," in *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1–5, 2018.

[13] A. M. Chiarelli, P. Croce, A. Merla, and F. Zappasodi, "Deep learning for hybrid eeg-fnirs brain–computer interface: application to motor imagery classification," *Journal of neural engineering*, vol. 15, no. 3, p. 036028, 2018.

[14] D. Achanccaray, C. Flores, C. Fonseca, and J. Andreu-Perez, "A p300-based brain computer interface for smart home interaction through an anfis ensemble," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–5, IEEE, 2017.

[15] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[16] H. Yin, "The self-organizing maps: background, theories, extensions and applications," in *Computational intelligence: A compendium*, pp. 715–762, Springer, 2008.

[17] P. H. Winston, *Artificial Intelligence (3rd Ed.)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1992.

[18] M. F. Møller, *A scaled conjugate gradient algorithm for fast supervised learning*. Aarhus University, Computer Science Department, 1990.

[19] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[20] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[21] G. E. Hinton, "A practical guide to training restricted boltzmann machines," in *Neural networks: Tricks of the trade*, pp. 599–619, Springer, 2012.

[22] A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE transactions on audio, speech, and language processing*, vol. 20, no. 1, pp. 14–22, 2011.

[23] M. Tanaka and M. Okutomi, "A novel inference of a restricted boltzmann machine," in *2014 22nd International Conference on Pattern Recognition*, pp. 1526–1531, IEEE, 2014.

[24] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–48, Springer, 2012.

[25] S. A. Cortez, C. Flores, and J. Andreu-Perez, "Single-trial p300 classification using deep belief networks for a bci system," in *2020 IEEE XXVII International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, p. to appear, IEEE, 2020.

[26] S. A. Cortez, C. Flores, and J. Andreu-Perez, "A smart home control prototype using a p300-based brain-computer interface for post-stroke patients," in *Smart Innovation, Systems and Technologies*, p. to appear, Springer Nature, 2020.