Under-sampling and Classification of P300 Single-Trials using Self-Organized Maps and Deep Neural Networks for a Speller BCI

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 control machines or other devices by translating its brain activity and using it as commands. This kind of technology has as potential users people with motor disabilities since it would allow them to interact with their environment without using their peripheral nerves, helping them to regain their lost autonomy. One of the most successful BCI applications is the P300-based Speller. Its operation depends entirely on its capacity to identify and discriminate the presence of the P300 potentials from electroencephalographic (EEG) signals. For the system to do this correctly, it is necessary to choose an adequate classifier and train it with a balanced data-set. However, due to the use of an oddball paradigm to elicit the P300 potential, only unbalanced data-sets can be obtained. This paper focuses on the training stage of two classifiers, a deep feedforward network (DFN) and a deep belief network (DBN), to be used in a P300-based BCI. The data-sets obtained from healthy subjects and post-stroke victims were pre-processed and then balanced using a Self-Organizing Maps-based under-sampling approach prior training looking to increase the accuracy of the classifiers. We compared the results with our previous works and observed an increase of 7% in classification accuracy for the most critical subject. The DFN achieved a maximum classification accuracy of 93.29% for a post-stroke subject and 93.60% for a healthy one.**

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

I. Introduction

A brain-computer interface (BCI) is a technology that uses people's brain signals to grant them control over machines [1]. The brain activity is recorded using different brain imaging techniques, such as electroencephalography (EEG) and magnetic resonance imaging (MRI). EEG is usually employed in many applications due to its non-invasive character and low cost compared to other techniques. Paradigms such as oddball and motor imagery (MI) are used to elicit well defined physiological responses that can be recorded through EEG and then interpreted by the BCI system for a specific application [2]. Motor-impaired people would benefit tremendously from the BCI technology since it would provide them with new means to interact with their surroundings, improving their lifestyle [3].

The *Speller* is a well-known BCI application first proposed by [4] that uses the P300 waveform, which is an endogenous event-related potential (ERP) triggered by visual stimuli and recorded using EEG. This BCI acts as an alternative communication tool in which the users can write characters in a computer using their induced P300 responses. An oddball paradigm is commonly used to elicit the P300 waveform and it consists of presenting the user target stimuli blended among irrelevant stimuli. 300ms after the target stimulus was presented, a positive potential (taking the name P300) can be spotted in the user's EEG signals. The simplicity of this paradigm makes the BCI's classifier require fewer samples for training in comparison with others, making it useful for developing quick solutions [5].

However, the main drawback of using this paradigm is that it provides unbalanced data-sets. Unbalanced distributions affect negatively the classifier's training because the resulting model would tend to classify only the majority class. There have been several works focused on how to deal with this kind of data-sets and also on how to balance them. The work of [6] presented methods for balancing two-class data-sets looking to improve the classification accuracy for the minority class using an under-sampling approach (discarding samples from the majority class). The authors used the *k-means* algorithm to cluster both the majority and minority classes and selected the clusters that best represent the data-set under a specific criterion. They compared their approaches with other methods and showed an increase in classification accuracy when using a neural network for the classification of different data-sets. In [7], the authors used Self-Organizing Maps (SOMs) also for under-sampling. Using two SOMs, they projected all the samples in the data-set into a 2-D space expecting two regions to be defined for each class. The samples that lay in-between these regions were removed until the set was balanced. They also reported an increase in classification accuracy in comparison with random under-sampling methods.

Machine learning algorithms have been successfully used for developing classifiers that identify and discriminate user's commands automatically. These classifiers are of special importance since their performance will determine the BCI's correct functioning. The work of Hoffmann [5] used Bayesian Linear Discriminant Analysis (BLDA) and Fisher's Linear Discriminant Analysis (FLDA) to classify the P300 potential of five disable subjects and four healthy subjects. They were able to achieve a classification accuracy of 100% for all the subjects using multiple blocks for classification instead of single-trials. The main disadvantage of classical machine learning algorithms, like Support Vector Machines (SVM), is that they often require feature reduction pre-processing stages to achieve a good classification accuracy [8] [9]. 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, which means raw data can be fed to them directly. This is possible because its learning scheme allows them to identify important features automatically. Regarding deep learning techniques for BCI applications, in [10] the authors used a convolutional neural network (CNN) for P300 classification with which they were able to achieve an accuracy of 95.5% using target by block. In [11] the authors used instead a deep belief network (DBN) and reported a classification accuracy 86.4%. In the work of [12], a deep feedforward network (DFN) was used to classify MI data obtained with EEG and functional near-infrared spectroscopy (fNIRS). They reported a maximum classification accuracy of 94.6%. These last two neural networks have the advantage of having a shorter training period in comparison with the CNN, making them suitable to test different configurations and analyze their results faster.

In this paper, we propose an approach to train a classifier to be used in a P300-based BCI for stroke victims. We deal with the data-set unbalanced distribution using a Self-Organizing Map-based under-sampling approach. Two deep neural network architectures are proposed for P300 single-trial classification and their performance is compared 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 volunteered 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 three stroke victims. 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 also informed about the academic objectives pursued in this work and ensured the preservation of their anonymity.

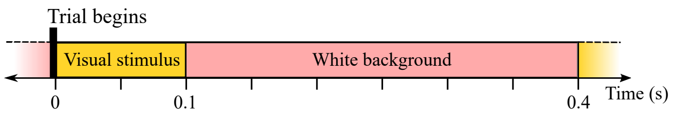

Fig. 1: Protocol's time scheme. One of the six images (visual cues) was randomly selected and displayed on the screen for 100ms followed by a white background for 300ms. After displaying all the six images once, the process repeats itself between 20 and 25 times.

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.

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

B. Experimental Setup

The protocol used in this study was based on Hoffmann's work [5]. Six different images, each one representing an action the subject would like to carry out, were randomly flashed on a screen continuously. The Fig. 1 shows the timing scheme of the experiment. The time interval in which the six images are flashed once is called a block. Between 20 and 25 blocks make up a run and, six of these form a session. Four sessions, recorded over two days, were obtained from the nine subjects. Through all the experiment, the participants were asked to count how many times the image they were told to pay attention to appeared on the screen.

C. Signal Pre-processing

The EEG signals were downsampled by a factor of 20 and any spurious spectral components were removed using notch filters. Signals then passed through a sixth order Butterworth filter with cut-off frequencies in 1 and 15 Hz. For each electrode, the data points recorded in one second right after an image was presented on the screen were extracted and stored in a 16-by-120 array, defining a trial. Fig. 1 shows the beginning of a trial. Artifacts and outliers in trials were eliminated by winsorization [5]. Finally, signals on each trial were standardized.

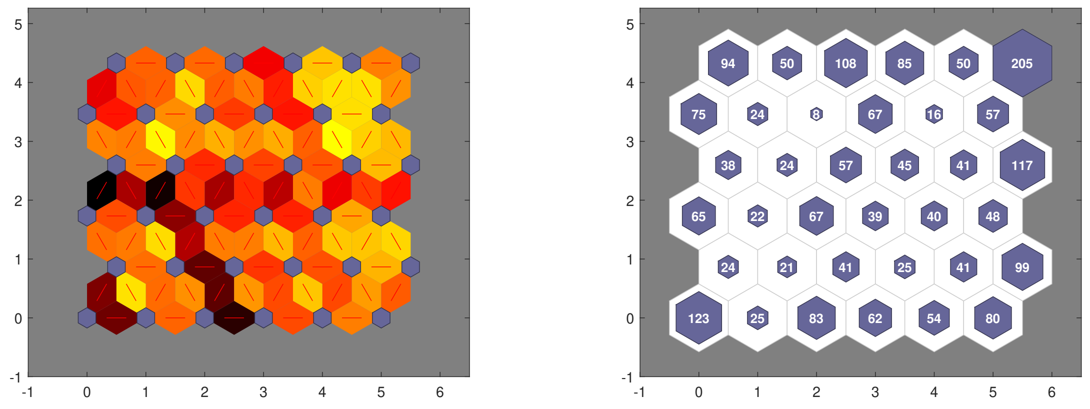

Fig. 2: Self-organized map of the subject S01 *non-target* trials. Fig. 2a shows the neurons (blue hexagons) and the distances (red lines) between each in the hexagonal grid. The darker colors indicate a larger distance while the lighter ones, the opposite. A band of dark segments can be spotted from the lower center region to the left center region, separating the *non-target* trials in two groups. Fig. 2b shows the number of hits for each neuron. The neurons on the right upper corner were selected since the distance between these was small (indicating same characteristics) and their hits added up to the correct amount of observations needed.

D. *Feature Vectors and Balancing Process*

Feature vectors were constructed rearranging the subject's trials. The sample points of each channel were first demeaned and then concatenated forming a row vector such as:

$$\boldsymbol{I} = \begin{bmatrix} S_1^1 & S_1^2 & S_1^3 & \dots & S_1^{16} & S_2^1 & S_2^2 & \dots & S_{120}^{16} \end{bmatrix},$$

wherein for a single data point S_k^i, i indicates the channel it belongs and k its position with respect of time. Each trial and thus, each feature vector, has a label which indicates whether or not its related visual stimulus triggered a P300 response. If that is the case, the trials are called *target*, and when not, *non-target*. However, since consecutive trials overlap, the adjacent *non-target* trials to the *target* ones will also present the P300 waveform at some degree. These kind of trials were removed before applying the balancing process.

The uneven amount of classes in the subject's data due to the experiment protocol was balanced using Self-Organizing Maps (SOMs). Proposed by Kohonen [13], SOMs are unsupervised machine learning algorithms focused mainly on clustering and pattern recognition tasks. Basically, these shallow neural networks perform a topological mapping of high dimensional inputs onto a discreet 2-D space for easier characteristics analysis and visualization. The neurons 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 [14]. 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 hexagonal 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 location vector on the grid. The weight vectors have the same dimension as the SOM input and their values are initialized with small random numbers. The training process is divided into two parts, which are repeated a defined amount of times or until convergence. First, a winning neuron is selected by comparing the euclidean distance between all weight vectors and a given certain observation $\boldsymbol{x}(t)$:

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

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

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

where $\alpha(t)$ is the so-called learning rate parameter and $\rho(c,j,t)$ is the neighborhood function which simulates the lateral interconnections between the winning neuron and its neighbors. The learning rate is usually defined as a monotonically decreasing linear function whose values are greater than 0 and less than 1. The neighborhood function is the Gaussian $\rho(c,j,t) = \exp\left(-\frac{\|\boldsymbol{r}_c - \boldsymbol{r}_j\|^2}{2\sigma^2(t)}\right)$, where σ is the neighborhood radius and depends on how the neurons are arranged (rectangular or hexagonal grid). The Eq. 2 essentially shift the weight vectors of the surrounding neurons towards the winning one in a proportional manner, making them more likely to recognize similar patterns 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 data patterns.

Only the *non-target* trials were fed into a SOM network with the purpose of choosing those with the most alike characteristics. The SOM had a 6-by-6 hexagonal grid and its training process was fixed for 50 epochs. The trials were chosen following the steps shown in Algorithm 1. Essentially, this algorithm group neurons highly related (small distance between them) trying to match their hits with the number of *target* trials (Fig. 2b). The parameters d_e and β were empirically tuned for each subject considering the results of its trained SOM. Ideally, the distance between all neurons in the grid should be small since the inputs are all same class trials; but, as shown in Fig. 2a, separations can be spotted suggesting differences in the information each trial carries. The resulting vectors were stacked together with the *target* ones in a random order, shaping the training and testing matrices.

Algorithm 1 Clusters selection

```
1. Compute the $k$ euclidean distances, $k$ ranging from 1 to
   85, between each adjacent pair of neurons $(n_y, n_z)$ in the
   grid graph:
       $\left\| \boldsymbol{w}_{n_y} - \boldsymbol{w}_{n_z} \right\| = d_k$
2. Link and group the neurons that are close to each other
   by comparing their distances with a threshold $d_e(t)$. For
   each $k$, if $d_k \leq d_e(t)$, then its neurons associated are
   joined.
3. The total number of hits in each group (or subset) $Gr_F$
   is extracted and compared with the total number of target
   trials.
       $\sum_{h \in Gr_F} hits(n_h) \approx \textbf{targets}.$
4. Considering a maximum error of 4 trials, if one group
   match, then it is selected for training. If there is more
   than one group satisfying that rule, then the one with
   the smallest distances on average between its neurons is
   selected. Any overflow in hits is trimmed if needed.
5. If there are no groups or the group's hits are not enough,
   the threshold is updated following
       $d_e(t+1) = \beta + d_e(t),$
   and the process is repeated from 2. until a $Gr_F$ match.
```

E. Classification

Two deep neural networks, a deep feed-forward network (DFN) and a deep belief network (DBN), were used separately to classify the subject's trials by their class. Neural networks can be seen as a function approximators [15], in which their internal parameters are tuned trying to minimize the error between actual outputs and desired outputs for specific inputs. Each neuron in the network is able to learn data patterns on different levels of abstraction (each layer), making the overall model capable of recognizing and classifying new inputs with similar characteristics to the samples used in its training. A DFN typically has multiple hidden layers and it is trained using back-propagation. Considering fully connected layers and differentiable transfer functions, the DFN training consists on minimizing a performance function using a gradient method. That is, taking the performance derivative with respect to the network's weights and bias terms and following the gradient (gradient descent) [16]. The network is updated with each one of the training samples until optimal performance is achieved or a specific number of epochs has passed.

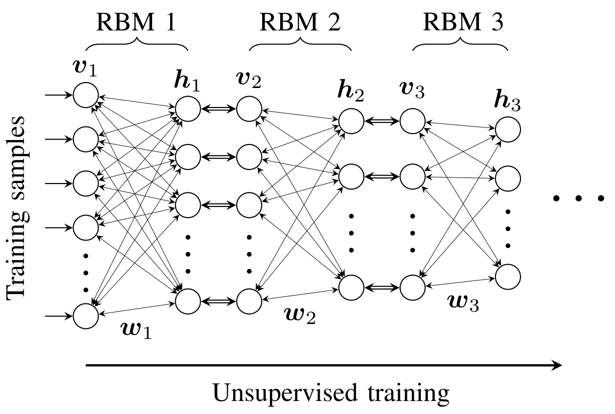

Fig. 3: DBN pretraining. The RBMs are trained in order, the hidden layers will serve as visible layers for the adjacent RBMs. When the pre-training is done, a classification layer is added and is fine-tuned in a supervised way.

A DBN is made of two or more restricted Boltzmann machines (RBMs) stacked on top of each other [17]. A RBM is a simple neural net with an input (or visible) layer fully connected to a single hidden layer. The DBN training is divided in two stages: the first one is the unsupervised training (or pretraining) of each RBM and the second stage is the supervised training of the whole network. The pretraining stage is motivated by the problems regular training by back-progation has. Specifically, the highly non-linear character of the network makes the minimization of its performance function by the gradient descent method troublesome due to been non-convex. The unsupervised greedy layer-wise training initialize the network's parameters to reduce the possibilities of the gradient getting stuck in local maxima (or minima) when the supervised training stage takes place. In the unsupervised training stage, each RBM identify the most relevant characteristics of their respective inputs using the contrastive divergence (CD) algorithm [18]. A RBM's hidden layer will act as the input layer for the next RBM once its training is complete, Fig. 3 shows an schematic of this training process and how the RBMs are stacked forming the DBN. Once all the RBMs have been trained, the whole network can be fine-tuned using error back-propagation. The simplified unsupervised training algorithm is briefly presented below.

Let us assume a binary RBM for simplicity, which then can be used to generalize a model for real value inputs [19]. Let $\boldsymbol{v}$ and $\boldsymbol{h}$ 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 distribution for every possible joint configuration using the energy function is

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

where $Z = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$ is the partition function. The probability the neural net assigns to an input $\boldsymbol{v}$ is computed by summing all its hidden vectors, resulting in

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

A RBM can be assigned to a specific input by modifying its parameters. The derivative of the log probability of an input vector with respect to the RBM's weights can be used to define the 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

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

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

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

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 on each epoch, the inputs using only the hidden layer states by adjusting the net'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 [20].

The architecture proposed here for the two networks used is shown in Fig. 4. Both have four fully connected hidden layers of 60, 40, 30 and, 20 neurons on each respectively but different output/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. Any feature vector entering the network was first re-normalize from -1 to 1 automatically. The DFN was trained using the scaled conjugate gradient algorithm. 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 four minutes using a GPU.

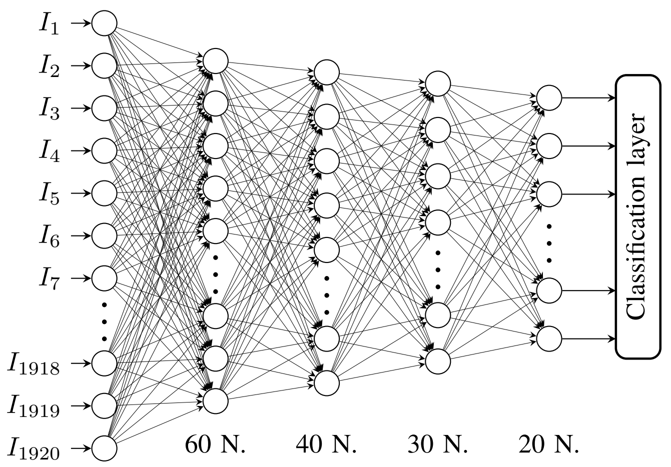

Fig. 4: Architecture of the neural networks proposed for P300 classification. The classification layer for the DFN is a single neuron and, for the DBN, are two.

For the DBN: four RBMs were trained and then 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 feature vectors entering the network were re-normalize from 0 to 1 automatically. 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 [21]. The training was fixed for 100 iterations and took approximately ten minutes per subject. Two neurons with a softmax activation function were used for classification. The DBN supervised training process was similar to the DFN one. A Nvidia GTX 1050 GPU and an Intel core i7 CPU were used for the calculations on MATLAB R2019a.

III. Results and Discussion

For each model, Table II shows the cross-validated classification accuracy and its standard deviation using 5-folds. The DFN achieved a classification accuracy higher than 91% for the subjects S01, S03, and S08. Only for subject S02, the DBN obtained better results than the DFN. It seemed the activation functions were important factors that determined the differences in accuracy between the two models. According to the analysis of [22], the hyperbolic tangent (tanh) function converges faster than the logistic sigmoid, thus optimizing the training stage. Specifically, the tanh function provides stronger gradients because its derivative ranges from 0 to 1, in comparison to the logistic sigmoid, which ranges from 0 to 0.25. The tanh function also produces outputs that are on average zero (since its range goes from -1 to 1), making the training process by a gradient method efficient and less biased. Even with the pre-training stage, the DBN was not able to match the DFN, although the difference was not massive.

In both classifiers, the performance for each subject was above 80%, except for subject S07. The critical medical condition of that subject may have negatively affected the classifiers' training and testing process, resulting in lower performance with respect to the others. Even though the P300 is an endogenous brain response that can be elicited without problems in post-stroke victims, the subject's concentration plays a fundamental role that will determine overall if the P300 potential is evoked or not. The results suggest subject S07 concentration decreased over time possibly due to fatigue. The subjects S08 and S09, both post-stroke patients, obtained even a better performance than most of the healthy subjects, which clearly indicates that a p300-based BCI for ischemic stroke victims using this classifier will work correctly.

Table III shows the results of our previous works [23] [24]. The classification models: a multi-layer perceptron (MLP), a support vector machine (SVM) and, an ANFIS ensemble were also validated using a 5-fold cross-validation method. Their training process used a balanced data-set obtained through a random under-sampling method. The ANFIS classifier was not tested with subject S05 because, for that moment, he was still in the process of having its EEG signals recorded. The DFN outperformed all these classifiers for each subject, especially with subject S07. Although the MLP had similar architecture to the DFN, for some subjects, their results were not even close. This reckons how the trial selection in the balancing process can heavily influence the classifier's training stage and performance. We stated the trials were recorded under same circumstances, however differences between them due to artifacts generated by subject's fatigue or external events are likely to happen. The cluster-based balancing process using SOMs proposed here aimed to select *non-target* trials that contained less artifacts and were more alike in terms of their characteristics. Such trials should represent the majority of their class, assuming the subject's concentration was somewhat maintained and there were few external disturbances. We hypothesized the larger and closer clusters in the SOM map after its training would contain these kind of trials while the rest would scatter around them. With strong differences between the two classes, the classifier would have less trouble discriminating them. There is a possibility the clusters in the SOM cannot be easily separated or differentiated, indicating overall similar trials, in which a random selection would yield similar results. However, this would usually happen in healthy subjects and in very good conditions. Let us not forget this BCI was designed targeting post-stroke victims which, due to their health condition, may present problems and require special measures that are not usually contemplated. The improvement in the performance of subjects S07 an S08 supports the importance of a balancing process under certain criteria rather than randomly.

TABLE II: Cross-validated model accuracy

Subject	Classification models	
	SOM + DFN	SOM + DBN
S01	93.60 (0.7)	90.13 (1.6)
S02	81.38 (2.3)	83.48 (1.5)
S03	92.09 (1.6)	86.28 (3.2)
S04	83.24 (2.4)	81.00 (0.9)
S05	90.09 (2.0)	84.20 (3.7)
S06	84.80 (2.2)	81.44 (2.2)
S07	77.70 (3.8)	72.04 (1.6)
S08	93.29 (1.4)	88.40 (1.5)
S09	86.60 (2.7)	84.00 (2.3)

The two models presented here were designed ultimately to classify trials by block, as in the BCI systems proposed by [5] [10]. Their single-trial classification accuracy would allow the system to reduce dramatically the number of blocks needed to be certain about the user's commands, resulting in a 100% classification accuracy in less time. This is of special importance since the BCI will eventually help its user to take actions that depend on quick responses, like answering its phone.

TABLE III: Previous classification results

Subject	MLP [24]	SVM [24]	ANFIS [23]
S01	91.8	91.5	85.3
S02	80.3	79.1	77.4
S03	85.3	83.9	79.3
S04	75.7	78.9	79.5
S05	84.9	83.4	-
S06	83.0	81.7	72.4
S07	68.6	69.2	70.1
S08	89.6	85.5	74.9
S09	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 for 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 fatigue from diminishing their performance. An alternative approach would be to reduce 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 from the subject.

Both networks not required computing-intensive pre-processing stages. Their deep architecture allowed them to learn and form decision rules efficiently basically from raw data. Small variations in their architecture, specifically classification layers and activation functions, determined overall the differences in their performances. The balancing process based on SOMs to which the subject's data was put through was a crucial stage that enabled the classifiers to achieve a higher classification accuracy. The SOMs clustered the *non-target* trials and made possible their selection under a certain criterion. The results obtained here outperformed our previous works. The most critical subject (S07) in the patient cohort improved its performance by 7%.

As future work, we intend to include amyotrophic lateral sclerosis (ALS) patients and also test different machine learning algorithms.

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] 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.

[3] 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.

[4] 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.

[5] 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.

[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] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

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

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

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

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

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.