Error perception classification in Brain-Computer interfaces using Convolutional Neural Networks

José Rafael Cabral Correia
jose.r.c.correia@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

January 2021

Abstract

Motivation: Capturing the error perception of a human interacting with a Brain-Computer interface is a key element in improving the performance of these systems and making the interaction more seamless. Convolutional Neural Networks (CNN) have been recently applied for this task rendering the classification model free of feature-selection. Despite the advances in the last years, there is still room for improving the accuracy so that it can be used in real-life applications.
Objectives: The goals of the present work are to investigate, replicate and validate previous CNN models used for error perception classification; to propose new CNN models based on the advances in Machine Learning and lastly, to make all the code and models developed publicly available.
Methods: After a literature review, three recent CNN models for error-related potential (ErrP) classification are replicated. Then, the author evaluates different new models that result from investigating CNN models used for classification of ErrP and *P300* signals. The Monitoring Error-Related Potential dataset is used to train and test all the models.
Results: The best model from the literature achieves an accuracy, sensitivity, and specificity of 77.6%, 71.7%, and 83.1%, respectively. For the best proposed model, these metrics are of 80.4%, 75.9%, and 84.7%, respectively, which represents a statistically significant increase on the literature models ($p = 0.0004$). Furthermore, an EEG input with a shorter temporal size of $600ms$ is successfully applied instead of the typical one-second-long input without significant loss of performance ($p = 0.647$). All models are made available online for easier future replication and peer review.
Conclusions: The new proposed model outperforms the state-of-the-art. The $600ms$ input allows faster processing times in real-time BCI applications without loss of performance.
Keywords: Brain-computer interface; Convolutional Neural Networks; Feedback error

1. Introduction

The development of Brain-Computer Interface (BCI) systems is a very active field of research and has grown considerably during the last decades [1]. Usually, computer input requires the user to perform a muscular action controlled by the brain such as when using the mouse, keyboard, voice commands, or others. BCI systems define a way of interaction between a human and a computer that relies solely on brain activity, rendering the use of an intermediate actuation step by means of peripheral nerves and muscles unnecessary [2].

Different brain signals can be used to assess the subject's intention so that it is translated into a computer or machine command. One such signal is the error-related potential (ErrP) which has gained popularity among the BCI community but still presents low accuracy rates when compared to its counterparts [3]. Error-related potentials provide insight as to when a user makes or perceives an error during the execution of a task [4]. If high accuracy on ErrP classification is obtained, BCI systems can better predict the real intent of the user by automatically detecting errors and correct for them, thus increasing the system's performance.

Many methods can be used to classify the occurrence of ErrPs in the brain. However, issues such as the processing of large amounts of data or lack of generalization present challenging problems. In the last few years, Deep Learning (DL) and, in particular, Convolutional Neural Networks (CNN) have become emergent technologies in tackling these problems. Deep Learning brings a lot of advantages when handling complex and large types of data and it is referred to as state-of-the-art in many fields such as image recognition [6], natural language [7], stock market [8], advertising [9] or healthcare [10]. It allows the scientific community to move away from feature engineered meth-

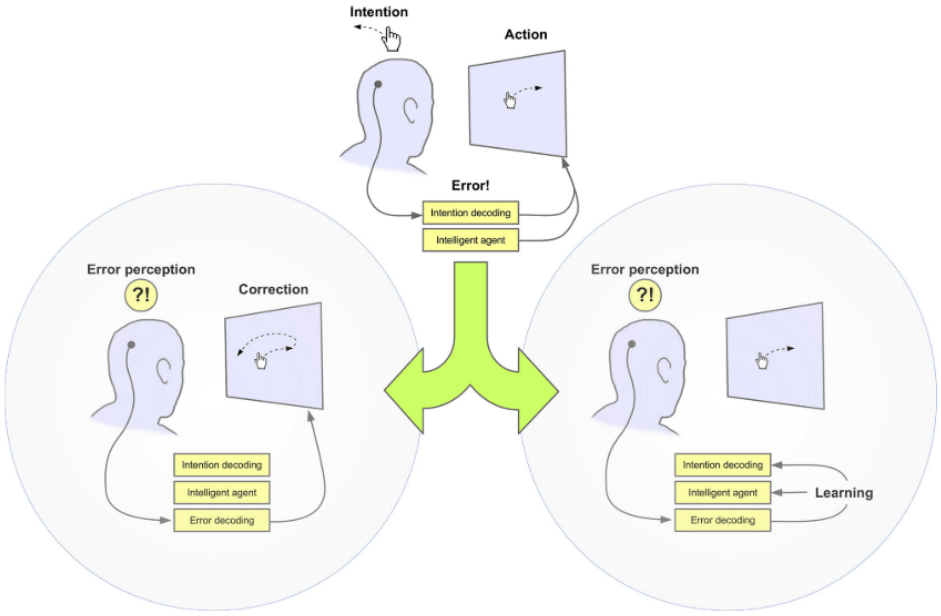

Figure 1: Using error-related potentials to improve BCI systems. **Top:** BCI system mis-classifies the intent of the user and feedback is provided. **Left:** Error is perceived by the user and the single-trial detection of the error-related potential is used to correct the erroneous action. **Right:** Alternatively, the detected ErrP can be used to improve the system's classifier. Taken from [5].

ods and provides end-to-end solutions that learn meaningful, high-level features on their own [11].

The present work has three main goals:

1. investigate and replicate previous CNN models used for ErrP classification;
2. develop new models based on advances in the field of Deep Learning that can provide improvements to the classification of ErrP;
3. provide open-source code for all the models reviewed in this work and the new proposed ones.

2. Background

This section presents some background to the problem and the state-of-the-art.

2.1. Brain-computer interface

Conventionally, BCIs are composed of five components: acquisition, pre-processing, feature extraction, classification, and application [12]. During acquisition, the electrical activity of the brain is registered and converted into digital information. Electroencephalography (EEG) is a very common and cheap acquisition technique used for this purpose. The pre-processing stage removes noise from the signal, thus increasing its signal-to-noise ratio, or selects the relevant parts of the signal to be processed. Feature extraction finds relevant characteristics in the signal which are fed into the classifier that then associates the features with one of several possible application commands (classes). Finally, the application receives the intent of the user as a command, applies it, and may or may not display some feedback to the user.

Although this is the conventional scheme, the distinction between feature extraction and classification is only applicable for BCI systems that do not use DL methods to train their models. Deep Learning algorithms perform those two steps at the same time: non-engineered features are learned while the classifier is being trained.

2.2. Event-related potentials

Event-related potentials (ERPs) are electrical brain signals elicited by a stimulus that may have different origins: sensory (visual, auditory, tactile, etc), cognitive (attention, memory, perception, etc), or motor. ERPs are composed of one or more wave components which are characterized by their amplitude, latency, and scalp distribution. Each component is commonly named in the literature according to its polarity (negative or positive) and latency (in milliseconds): *P50, N100, P100, N200*, and so on. A very common ERP component is the *P300* which is elicited around the parietal cortex after a rare and task-related stimulus is presented in the middle of a random series of stimulus events [13]. It has been extensively studied and applied in the field of BCI [14].

The present work focuses on Error-related potentials (ErrPs), yet another ERP component that has been much less studied than *P300* but hold promising advances in the field of BCI. Nowadays, ErrP has become an umbrella term in the more engineering-oriented research referring to the var-

ious sub-components that may be correlated to error handling in different paradigms.

Various studies have tried to use the occurrence of such signals to detect the error perception of the subject when using a BCI [15]. The signal is elicited when the user, after being presented with a feedback stimulus, realizes a mistake was committed by the system when trying to identify the users' intention. The ErrP presents three main features in its waveform: a positive peak at around $200ms$ after feedback, a large negative deflection between 200 and $250ms$ and another positive peak at about $320ms$ [5]. This pattern is generated in the anterior cingulate cortex (ACC) [5] and has been reported to be more or less constant even when comparing trials months apart [16].

An advantageous feature of ErrP is that it is a physiological signal that occurs naturally during the interaction with Brain-Computer interfaces, contrary to other ERPs that require training or stimulation by the user such as those elicited with a motor imagery paradigm.

2.3. Convolutional Neural Network

CNN is a family of deep learning models that makes use of the convolution mathematical operation to extract meaningful features from the data and combine them in order to generate the intended output such as the classification of an image or a time series.

Convolutional layers extract useful features by sliding filter tensors (kernels) through the data. Depending on the traversed dimension (considering a 2D matrix whose two dimensions are space and time), the result of a convolution is a set of spatial and/or temporal features. To add non-linearity to the model many different activation functions can be used. The sigmoid and the hyperbolic tangent (TanH) are very common ones but suffer from the vanishing gradient problem [17]. An extensively used function that avoids this problem is the rectified linear unit (ReLU). This function, however, suffers from the 'dead' neuron problem, where the null derivatives computed during backpropagation prevent weights and biases from updating during training [17]. To address this behavior, both the LeakyReLU and the exponential linear unit (ELU) can be used. Another advantage of these two functions is that they can produce negative outputs, allowing updates to the weights and the biases in both directions.

After the convolutions and non-linearities, at the end of a CNN model there is usually one or more fully connected (FC) layers working as a regular artificial neural network combining the extracted features.

A common problem that emerges in models where a huge number of parameters are estimated is overfitting of the data. The model may predict very well the cases it was given to train but predicts poorly new unseen cases (test set). Two ways to attenuate this effect are using batch normalization (BN) and dropout layers [3]. Dropout layers zero-out a certain percentage (dropout rate) of nodes before feeding the data to the next layer.

2.4. State-of-the-art

Several models have been proposed over the years to classify different ERPs using both classical and deep learning approaches. Recently, focus has been given to the latter due to their high accuracy and the fact that the model performs both the feature extraction and classification tasks, meaning that no *a priori* optimal feature search is needed. In this work, the author focuses on three recent studies that use CNN models for ErrP classification which are later used for comparing with the new proposed models. Furthermore, CNN models that classify P300 signals are also considered. Since these have been more extensively investigated, they provide insights into the model's architecture and on how to improve their performance.

In 2018, Luo *et al.* [18] proposed a model (*CNN-L*) which applies temporal convolution in the first layer and then applies spatial convolution. After the convolutions, the model applies batch normalization, average-pooling, and a 50% rate dropout before the fully connected layer that outputs two-class nodes. This particular ErrP classification model was used to improve the efficiency of experiments that aimed at obtaining human intuitive preferences. By observing the error perception caused by a random selection of the computer (contrary to the user preference) the authors were able to train the model to reach a 67% accuracy level.

In the same year, Torres *et al.* [19] proposed a model that convolves both spatial and temporal dimension at the same layer. Their model, called *ConvNet*, performs a mixed convolution with a max-pooling layer followed by another mixed convolution with max-pooling and a FC layer at the end with two nodes as output. Additionally, as part of the pre-processing before feeding the data into the *ConvNet* model, the EEG passes through three stages: artifact removal, ZCA whitening, and cropping. During the last stage, the signal, which has a size of 64×563, is cropped to a size of 64×64 at a random point, decreasing the temporal dimension of the input. The authors argue that this process decreases the likelihood of trapping the model at a local minimum during training.

In 2019, Bellary and Conrad [3] described a CNN model called *ConvArch* that only takes two electrodes as input after visual inspection of the topo-

graphical maps of the averaged ErrP. The largest variation was observed at the region of the ACC, hence the choice of the *FCz* and *Cz* electrodes out of the 64 available channels. Again, the first convolution layer performs a mixed convolution operation with both the temporal and spatial dimensions. Then, a set of three modules follow, each being composed of a temporal convolution layer (with ReLU activation functions) and a max-pooling layer that halves the size of the matrix data. In the end, a single FC layer is followed by a softmax activation function that outputs two nodes. The authors experimented with two versions of this architecture: one as just described (*ConvArch1*) and another with added BN and dropout layers (*ConvArch2*). Unfortunately, the exact arrangement of these added layers inside the architecture of *ConvArch2* and the dropout rates are not mentioned in the paper and must be defined through an educated guess.

The comparison of results from the three models presented is not obvious due to the different datasets, pre-processing, and training methods used. In the next section, these stages are uniformized as much as possible, allowing an easier comparison of the performances.

3. Methodology

This section presents the dataset used and the proposed models to be tested against the literature models.

3.1. Dataset

The used dataset was created by Chavarriaga et al. [20] and is publicly available [21] at the BNCI Horizon 2020 project website under the name *Monitoring error-related potentials*. The signals are generated while the user is monitoring an external device upon which no control is given. Each subject has to monitor a green square cursor which can travel along a horizontal line made up of 20 evenly spread positions. On each trial, a target appears either at the left or at the right of the cursor and the cursor moves in the direction of the target. After reaching the target, the cursor stops, and a new target comes up no further than 3 positions away from the last target. The subjects are informed that the goal of the cursor is to reach the target. To present erroneous behavior to the subject, the cursor has a 20% probability of moving away from the target, contrary to its determined goal. The dataset includes six subjects (mean age 27.83 $\pm$ 2.23 years) who performed two sessions each separated by several days. Each session contains 10 blocks (3 minutes each) with approximately 50 trials per block. The raw dataset contains 1030 target trials and 4085 non-target trials as refered on Table 1.

Table 1: Number of target (with ErrP) and non-target (without ErrP) trials in the dataset.

Trial type	Per subject						Total
	1	2	3	4	5	6	
Target	181	184	142	165	186	172	1030
Non-target	628	654	706	676	696	725	4085

3.2. Epoch window

An important consideration for future use in real-time applications is the temporal epoching window size, which is the temporal range from the EEG signal that is fed to the model as input. A larger input may allow the model to search for more features. However, if we consider the application of an error perception classifier running in real-time, then the temporal epoch window should be as short as possible to decrease the delay between the feedback presentation and the classification. Hence, a compromise must be achieved that both maximizes the useful information and minimizes the lag-time during online settings.

In this work, we hypothesize that $600ms$ after feedback presentation is enough to maintain a high level of accuracy while significantly reducing the lag-time. This epoch window keeps the waveform at the center of the interval as the ErrP occurs mainly at around $300ms$, including both its start and tail. Considering that the majority of the approaches use an epoch size of around one second [3, 19, 22] this would reduce the lag-time by about half, which is a significant improvement.

To confirm this hypothesis, an experiment is performed where a group of different ranges is used to epoch the input signal. The first range starts at feedback presentation and lasts $1000ms$. This is the most common range and is thus used as the control range. Then, five increasingly smaller ranges follow: from the range $[0, 600]ms$ until the range $[0, 200]ms$ in steps of $100ms$. After epoched, the data is fed to a CNN model, and the accuracies obtained are compared. If no significant differences in performance are observed between the control range and the $[0, 600]ms$ range, then the hypothesis is valid and the shorter window both preserves signal information and reduces lag time. The remaining ranges are tested to verify how much the epoch window can be reduced before any significant performance reduction is noticed.

3.3. CNN models

The mathematical abstraction of the problem considered can be broadly divided into three parts as shown in Figure 2: the input (EEG signal), the function (CNN model), and the output (classification classes). The goal in this section is to focus on the function and thus propose new CNN mod-

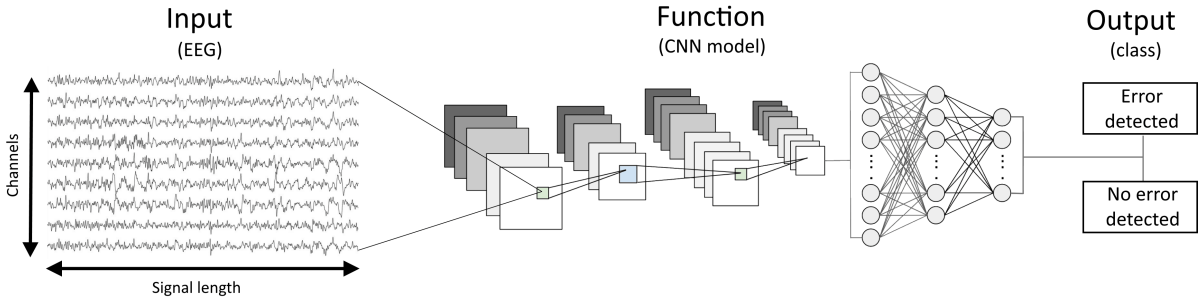

Figure 2: Processing pipeline when using a CNN model.

els able to classify, with the best accuracy possible, the EEG signal generated after a feedback is presented to a patient. This classification depends on the presence of a specific event-related potential called ErrP. If the ErrP waveform is present then the model should classify the given EEG segment as "containing an ErrP" or otherwise "not containing an ErrP". From the analysis of the CNN models used for *P300*, several characteristics can be drawn, which constitute improvements over previous ErrP classifiers.

Concerning input normalization and solely considering the three models focusing on ErrP, only *ConvNet* applies some kind of normalization to the input by applying ZCA whitening in the preprocessing stage. On the contrary, all *P300* models use some form of input normalization. While some apply normalization during the preprocessing stage, one model uses batch normalization [23], which makes the normalization part of the model's architecture. Hence, in this work, a normalization layer is also added as the first layer to all the proposed models. Besides being used to normalize the input, BN layers can also be used in convolution layers, just before the activation functions, to normalize the data that suffers distribution changes due to the convolution.

Another important aspect is the order in which the temporal and spatial convolution layers are placed inside the architecture. One study states that performing the spatial convolution before the temporal convolution prevents the CNN models from learning the temporal features well [11]. Because of that, they present a mixed convolution, where the kernel convolves both in the temporal and spatial dimensions simultaneously. All the ErrP models seem to already implement this strategy, either by applying a mixed convolution or by adding the spatial after the temporal as done in the *CNN-L* model. In this work, the temporal convolution is applied first, followed by the spatial convolution. This is done, instead of the mixed version, to decouple these two independent operations.

Concerning the activation function used for the convolution layers, ReLU is the most used function in the considered *P300* models. During the last years, it has dominated the field of Machine Learning, being the most used activation function in state-of-the-art solutions [17]. However, it introduces the dead neuron problem that affects the training process. Both the LeakyReLU and the ELU functions address this issue, although the latter provides better generalization and faster training [17]. Therefore, and following the *ConvNet* and *CNN-L* models, the ELU function is used in this work.

To decrease the architectural complexity, the data size used by the model, and the training time, the proposed models use a stride size equal to the kernel size. This prevents input overlap when calculating two consecutive kernel convolutions. Because this stride choice largely reduces the data dimensionality, no other methods such as max-pooling are used for that effect. A study by Springenberg *et al.* re-assessed the state-of-the-art concerning object detection, where the common pipeline is using a sequence of convolutions with max-pooling layers followed by FC layers. They found that the max-pooling layers can be replaced by a larger convolutional stride without loss of accuracy [24].

When dealing with a binary problem, such as classifying the presence of a specific ERP in an EEG signal, the tendency is to use 2 output nodes which are identified as each of the classes. This is in fact what is observed from the collected studies, where almost all studies used 2 output nodes. The *BN3* model is the only to use one single output node which defines the probability of a certain class being the correct one. Given a threshold value, the most probable class is then selected. Having a single output node for a binary case not only avoids redundancy, but it decreases by half the number of weights in the last FC layer. Therefore, in this work, only one output node is used.

The dropout layer is an important element in studies with relatively small datasets, preventing the model from overfitting during training. If too many nodes are ignored, then the model might learn slowly or not learn at all. On the contrary,

if the rate is too low, then overfitting might happen and the purpose of the layer is lost. In the paper where dropout layers were proposed, the authors used dropout rates of 50% [25]. A more recent study suggests that using lower dropout rates such as 20% is better [26]. In fact, the *P300* models that implemented this technique, chose a relatively low dropout rate (20% and 25%). To verify the effectiveness of this layer, three different dropout rates are tested: 0%, 20%, and 50%.

The number of FC layers used after the convolution layers can also be optimized. As the number of FC layers increases, the model has the ability to develop more complex relations between the features to infer the correct classification. However, most of the ErrP models use only one FC layer, while the *P300* models tend to use up to three layers. To verify the effectiveness of extra layers in the architecture, three different models are tested: with 1, 2, and 3 FC layers.

The kernel size used in the convolution layer is a parameter with a wide range of values in the literature. Considering that the minimum and maximum values are $40ms$ and $125ms$, the author decides to experiment with three different kernel sizes within this range: 20, 40, and 60 samples which correspond to temporal sizes of approximately $40ms$, $80ms$ and $120ms$, respectively.

The removal of the BN and ELU activation layers after the first convolution is also tested to verify if it produces higher accuracy.

All models are developed using the PyTorch library and trained with GPU acceleration. The literature models are trained with the parameters defined in the original papers and the proposed models use an SGD optimizer with a learning rate of 10^{-3}, a weight decay of 10^{-5}, momentum of 0.9 and batch normalization of 128.

The variations introduced here originate various models which can be seen in Table 4. The values inside square brackets present the best values for the specific features which are, at this point, still unknown and can only be asserted after running comparison tests in the next section.

4. Results

In this section, the results concerning the epoch window, the models from the literature, and the proposed models are presented and discussed.

4.1. Epoch window

Table 2 summarizes the results for the epoch window size. To compare each shorter epoch with the control epoch, t-tests are used with the accuracy as the comparison metric. Both the two first shorter ranges, $[0, 600]ms$ and $[0, 500]ms$, do not show accuracies statistically different from the control range, with *p-values* of 0.647, and 0.277, respectively. This means that the range can be decreased from the common $[0, 1000]ms$ range, to $[0, 600]ms$ or $[0, 500]ms$ without compromising the performance of the model. The remaining ranges, $[0, 400]ms$, $[0, 300]ms$, and $[0, 200]ms$, present a significant difference in accuracy from the range $[0, 1000]ms$, with *p-values* of 0.012, 0.001, and 2.92×10^{-11}, respectively.

Table 2: Accuracy of one model when training with different temporal windows (averaged over 5 independent training runs).

Epoch time range (ms)	Average	Sensitivity	Specificity
[0,1000]	79.1% ±0.5%	75.2% ±1.0%	82.9% ±1.0%
[0,600]	**79.4%** ±0.7%	**76.4%** ±2%	82.2% ±1.7%
[0,500]	78.5% ±1.0%	73.5% ±2.4%	83.1% ±3.1%
[0,400]	77.9% ±0.6%	74.2% ±2.6%	81.3% ±1.5%
[0,300]	73.3% ±1.5%	62.7% ±3.5%	**83.4%** ±1.8%
[0,200]	59.6% ±0.6%	47.4% ±1.6%	71.1% ±1.1%

The $[0, 600]ms$ epoch window provides the advantage of shortening the lag time when acquiring the input signal in real-time applications by about half. Despite the $[0, 500]ms$ range being shorter, the temporal difference of $100ms$ is not substantial and the decrease in sensitivity, although not statistically significant ($p = 0.109$), suggests that using the $[0, 600]ms$ range is a better approach. Hence, the hypothesis is experimentally verified and due to the advantages presented, the $[0, 600]ms$ epoching window is used for all the proposed models.

4.2. Literature models

Table 3 presents the performance results of the original and replicated models, where the accuracy, sensitivity, and specificity metrics are detailed.

After replicating the *ConvArch* model, it can be seen that its accuracy is 15% lower than that of the original paper. This difference is probably due to the dataset split which defines the test set used to evaluate the performance of the model. As a balancing technique, Bellary and Conrad replicated samples from the smaller class, as also done in the present work. If this process is not done randomly, then over-representation of a particular subject may emerge in the dataset. The results from Bellary and Conrad suggest that subject number 1 is over-represented in the test set. The problem with over-representing subject 1, in particular, is

Table 3: Performance of the original and replicated models from the ErrP literature. All the replicated results are averaged over 5 training runs with test sets randomly sampled from the second session of the dataset.

Author(s) [Year] Model	Results from	Acc.	Sens.	Spec.
Bellary et al. [2019] ConvArch	Original paper	86.1%	-	-
	Replicated	71.2% ± 1.7%	56.1% ± 6.7%	**85.4%** ± 3.2%
Torres et al. [2018] ConvNet	Original paper	-	77.5%	79.5%
	Replicated	51.7% ±0.8%	63.1% ±32.3%	40.9% ± 30.5%
	Replicated (no crop)	76.0% ±0.8%	67.1% ±3.8%	84.5% ±3.1%
Luo et al. [2018] CNN-L	Original paper	67.5%	-	-
	Replicated	**77.6%** ±0.7%	**71.7%** ±2.8%	83.1% ±1.5%

that it is the subject that always produces the best individual accuracy and thus, it introduces a bias by over-estimating the performance of the model. In the present work, however, the dataset split ensures the homogeneity of all the subjects for an unbiased performance evaluation.

The reproduced *ConvNet* model obtains an almost random accuracy of 51.7%. Both the obtained sensitivity (63.1%) and specificity (40.9%) present a standard deviation of around 30% and thus, it is clear that the model, when training, settles at a point where it more or less randomly classifies the majority of the samples as one of the two classes. The cropping of the input at the start of the architecture seems to be the problem and to test this hypothesis, another version of the same model was replicated, this time with no cropping of the input. This second model performs much better, with an accuracy of 76.0% and significant sensitivity and specificity (standard deviation of around 3%). When Torres *et al.* suggest the cropping step, they argue that papers such as that of Schirrmeister *et al.* [27] used it to reduce the probability of identifying a false training local minimum. However, the cropped percentage used by Schirrmeister *et al.* is of 50% (500 samples out of 1000), while the cropped percentage used by Torres *et al.* is of 11% (64 samples out of 563). Such a small crop makes the model blind to the overall temporal process, making it hard to correlate and extract features from different temporal crops, thus preventing the model from learning and performing well.

While the studies for the two previous models use the same dataset as the present work, the *CNN-L* model uses its own original dataset which contains 12 subjects. Therefore, the performance of the original and replicated models is not expected to be necessarily the same. In fact, an increase of 10% is achieved for the accuracy with the replicated model (77.6%) when compared with the original paper (67.5%). *CNN-L* presents the highest replicated accuracy of the three models. One factor that is singular to this model that may explain its accuracy is the sequential order of its convolutional operations. The majority of the models start with a spatial convolution followed by a temporal convolution or use a mixed convolution, where a *2D* kernel convolves both spatial and temporal dimensions simultaneously. *CNN-L*, on the other hand, places the temporal convolution before the spatial convolution which makes the model adequately learn temporal features [11].

4.3. Proposed models

In this final sub-section, the performance of the proposed models is analyzed and discussed. Table 4 summarizes the results and provides the actual parameters used by each model (inside square brackets). Each group of models used to evaluate the best parameter for a specific architectural component (kernel size, number of FC layers, presence of BN and ELU layers, and dropout rate) is compared with ANOVA and t-tests.

The first set of three models (1, 2, and 3 from Table 4) tests the kernel size for the possible values of 20, 40, and 60 samples. A one-way *ANOVA* reveals that there is no significant difference between their accuracies ($p = 0.808$). To avoid overcomplicating the model, when no statistical significance is found between a group of models, the simplest architecture is chosen. In the case of the kernel size, choosing a smaller kernel yields fewer parameters, which simplifies the model. Therefore, model 1 is chosen as champion, and models 4 to 9 implement a kernel size of 20 samples.

To analyze the appropriate number of FC layers, three models are used: model 1, 4, and 5 which all contain the same kernel size but a different number of FC layers (1, 2, and 3, respectively). Once more, a one-way *ANOVA* test does not find any statistical difference between these three models ($p = 0.568$). Therefore, the simplest model must be chosen. In this case, similarly to the previous one, fewer FC layers also generate less trainable parameters and since extra layers do not increase the performance of the model, they are not necessary. Following the same reasoning as before, model 1 remains the current champion, and models 6 to 9 implement a single FC layer in their architecture.

Concerning the presence of the BN layer and ELU activation function in the first convolution layer, models 1 and 6 are compared, since they only differ in that regard. A t-test shows that there is no statistical difference between the two models ($p = 0.117$). The simplest model, in this case, is model 6 which performs fewer calculations not

Table 4: Performance results for the proposed models.

Model number	Model architecture					Performance metrics		
	Kernel size	FC layers	BN & ELU	Dropout rate	Use BN	Acc.	Sens.	Spec.
1	20	1	Yes	0%	Yes	78.4% ±1.4%	72.6% ±1.8%	83.9% ±1.8%
2	40	1	Yes	0%	Yes	78% ±0.7%	71.7% ±1.3%	84.0% ±1.3%
3	60	1	Yes	0%	Yes	78.0% ±1.0%	71.2% ±1.9%	84.5% ±1.0%
4	[20]	2	Yes	0%	Yes	78.7% ±1.1%	73% ±3.6%	84.2% ±1.9%
5	[20]	3	Yes	0%	Yes	77.7% ±1.4%	71.9% ±1.8%	83.4% ±1.5%
6	[20]	[1]	No	0%	Yes	79.9% ±0.8%	75.3% ±1.5%	84.1% ±2.4%
7	[20]	[1]	[No]	20%	Yes	**80.4% ±0.7%**	**75.9% ±1.5%**	**84.7% ±1.5%**
8	[20]	[1]	[No]	50%	Yes	79.9% ±0.3%	77.4% ±0.5%	82.1% ±0.9%
9	[20]	[1]	[No]	[20%]	No	76.3% ±0.7%	68.9% ±1.7%	83.3% ±2.3%

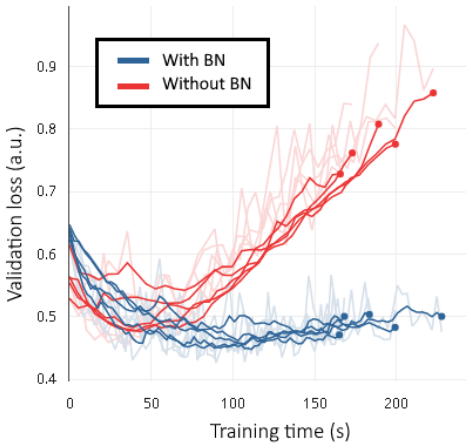

Figure 3: Loss plot for the validation set comparing the effect of models with and without batch normalization. The red group represents the models without batch normalization. Each line is an independent trained model.

considering the statistical metrics to normalize the batch or non-linear functions. This also simplifies the backpropagation calculations during training since fewer functions make part of the derivation chain. The non-linearity of the model is guaranteed with the activation function of the second convolution.

Next, the dropout rates are tested with models 6, 7, and 8, corresponding to rates of 0%, 20%, and 50%, respectively. The *ANOVA* test for these three models shows that there is no significant difference between their accuracies ($p = 0.427$). Since a dropout layer does not present any trainable parameters or performs calculations, there is no particular way to choose the most simple of the three models. In this case, the choice is based on two factors. Firstly, the advantage of having this layer for controlling the overfitting phenomenon excludes model 6 as it does not make use of it. Secondly, following the considerations of previous studies [26] and the practices of other CNN models, the smaller dropout rate of 20% is chosen. Therefore, model 7 is the new current champion.

Finally, model 9 is trained to verify the effectiveness of the BN layers in reducing the overfitting phenomenon. The comparison is performed between models 7 and 9. A t-test shows that the accuracies are statistically different ($p = 0.00005$). This shows that adding BN layers to the architecture, either at the output or before non-linearities, has an effective impact on the performance of the model. Furthermore, it does also have a positive influence in reducing the overfitting of a model during training. Figure 3 depicts the training evolution of the validation loss for several instances of model 7 (blue) and model 9 (red). From the graph, it is clear that, after the initial reduction of the error, both models display very different behaviors: the loss of model 7 remains more or less stable after the initial decrease, while the loss of model 9 starts to increase again which means that the model is overfitting. Thus, adding the BN layers effectively prevents overfitting.

5. Conclusions and future work

In this work, all three main goals proposed in the beginning are achieved. Three previous models from the literature were reviewed and replicated: *ConvArch*, *ConvNet* and *CNN-L*. Their results showed some incongruities concerning the results reported in the original papers, either due to the use of different datasets or to the insufficient architectural and training details.

To improve on the state-of-the-art, other CNN models were investigated, namely P300 classifiers. Since this type of signal is also an event-related potential, its nature is similar to that of an ErrP, and hence, the same type of Deep Learning architecture might produce good results. In fact, using multiple features from these models helped improve the accuracy of the new proposed models up to 80%, beating the state-of-the-art.

As future work, the author suggests using transfer learning to achieve a better generalization. First, a CNN model is pre-trained on a dataset with all subjects and later fine-tuned by freezing the early feature extraction layers and training with only one subject. This is expected to allow for both a good low-level feature extraction generalization and a good specificity for individual subjects. Additionally, more comprehensive datasets can be developed and publicly shared, with a higher number of subjects to study the effects of inter-subject differences in the generalization of models.

References

[1] Ujwal Chaudhary, Niels Birbaumer, and Ander Ramos-Murguialday. Brain–computer interfaces for communication and rehabilitation. *Nature Reviews Neurology*, 12(9):513–525, sep 2016.

[2] Aya Rezeika, Mihaly Benda, Piotr Stawicki, Felix Gembler, Abdul Saboor, and Ivan Volosyak. Brain–computer interface spellers: A review. *Brain Sciences*, 8(4), 2018.

[3] Sunny Arokia Swamy Bellary and James M. Conrad. Classification of Error Related Potentials using Convolutional Neural Networks. In *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 245–249. IEEE, jan 2019.

[4] Martin Spüler and Christian Niethammer. Error-related potentials during continuous feedback: Using EEG to detect errors of different type and severity. *Frontiers in Human Neuroscience*, 9(MAR):1–10, 2015.

[5] Ricardo Chavarriaga, Aleksander Sobolewski, and José del R. Millán. Errare machinale est: the use of error-related potentials in brain-machine interfaces. *Frontiers in Neuroscience*, 8(8 JUL):1–13, jul 2014.

[6] Zhong Qiu Zhao, Peng Zheng, Shou Tao Xu, and Xindong Wu. Object Detection with Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.

[7] Stephen Wu, Kirk Roberts, Surabhi Datta, Jingcheng Du, Zongcheng Ji, Yuqi Si, Sarvesh Soni, Qiong Wang, Qiang Wei, Yang Xiang, Bo Zhao, and Hua Xu. Deep learning in clinical natural language processing: a methodical review. *Journal of the American Medical Informatics Association : JAMIA*, 27(3):457–470, 2020.

[8] Eunsuk Chong, Chulwoo Han, and Frank C. Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187–205, 2017.

[9] Nehal Mohamed Ali, Marwa Mostafa Abd El Hamid, and Aliaa Youssif. Sentiment Analysis for Movies Reviews Dataset Using Deep Learning Models. *International Journal of Data Mining & Knowledge Management Process*, 09(03):19–27, 2019.

[10] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature Medicine*, 25(1):24–29, 2019.

[11] Hongchang Shan, Yu Liu, and Todor Stefanov. A Simple Convolutional Neural Network for Accurate P300 Detection and Character Spelling in Brain Computer Interface. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 1604–1610, California, jul 2018. International Joint Conferences on Artificial Intelligence Organization.

[12] Mariana Pedroso Branco. *Boosting BCI Technology*. PhD thesis, 2018.

[13] Tarik Al-ani and Dalila Trad. Intelligent and Biosensors. chapter 2. Intech, 2010.

[14] Rikvan Dinteren, Martijn Arns, Marijtje L.A. Jongsma, and Roy P.C. Kessels. P300 development across the lifespan: A systematic review and meta-analysis. *PLoS ONE*, 9(2), 2014.

[15] Martin Spüler, Michael Bensch, Sonja Kleih, Wolfgang Rosenstiel, Martin Bogdan, and Andrea Kübler. Online use of error-related potentials in healthy users and people with severe motor impairment increases performance of a P300-BCI. *Clinical Neurophysiology*, 123(7):1328–1337, 2012.

[16] Pierre W. Ferrez and José Del R. Millán. Error-related EEG potentials generated during simulated brain-computer interaction. *IEEE Transactions on Biomedical Engineering*, 55(3):923–929, 2008.

[17] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. pages 1–20, nov 2018.

[18] Tian-jian Luo, Ya-chao Fan, Ji-tu Lv, and Chang-le Zhou. Deep reinforcement learning from error-related potentials via an EEG-based brain-computer interface. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 697–701. IEEE, dec 2018.

[19] Juan M. Mayor Torres, Tessa Clarkson, Evgeny A. Stepanov, Christian C. Luhmann, Matthew D. Lerner, and Giuseppe Riccardi. Enhanced Error Decoding from Error-Related Potentials using Convolutional Neural Networks. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, volume 2018-July, pages 360–363. IEEE, jul 2018.

[20] Ricardo Chavarriaga and José del R. Millán. Learning From EEG Error-Related Potentials in Noninvasive Brain-Computer Interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(4):381–388, aug 2010.

[21] Data sets - BNCI Horizon 2020.

[22] Hongchang Shan, Yu Liu, and Todor Stefanov. *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*, volume 11730 of *Lecture Notes in Computer Science*. Springer International Publishing, Cham, 2019.

[23] Mingfei Liu, Wei Wu, Zhenghui Gu, Zhuliang Yu, FeiFei Qi, and Yuanqing Li. Deep learning based on Batch Normalization for P300 signal detection. *Neurocomputing*, 275:288–297, jan 2018.

[24] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, pages 1–14, dec 2014.

[25] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. pages 1–18, 2012.

[26] Sungheon Park and Nojun Kwak. Analysis on the Dropout Effect in Convolutional Neural Networks. volume 10112 of *Lecture Notes in Computer Science*, pages 189–204. Springer International Publishing, Cham, 2017.

[27] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for EEG decoding and visualization. *Human Brain Mapping*, 38(11):5391–5420, nov 2017.