2021 43rd Annual International Conference of the
IEEE Engineering in Medicine & Biology Society (EMBC)
Oct 31 - Nov 4, 2021. Virtual Conference

Error perception classification in Brain-Computer Interfaces using CNN

J. Rafael Correia[1] J. Miguel Sanches[1] and Luca Mainardi[2]

Abstract— **Capturing the error perception of a human interacting with a Brain-Computer Interface (BCI) is a key piece in improving the accuracy of these systems and making the interaction more seamless. Convolutional Neural Networks (CNN) have recently been applied for this task rendering the model free of feature-selection. We propose a new model with shorter temporal input trying to approximate its usability to that of a real-time BCI application. We evaluate and compare our model with some other recent CNN models using the Monitoring Error-Related Potential dataset, obtaining an accuracy of 80% with a sensitivity and specificity of 76% and 85%, respectively. These results outperform previous models. All models are made available online for reproduction and peer review.**

I. INTRODUCTION

BCIs are systems that make use of electrical activity from the brain to decode the user's intentionality and translate it into a task on a computer or machine.

One of the classes of electrical activity used by BCIs is the Event-related Potential (ERP) which is elicited as the result of a motor, sensory or cognitive stimulus. Among these events, the most commonly used in BCI systems are the P300 evoked potentials and Event-related Synchronization/Desynchronization (ERD/ERS) [1]. More recently, Error-related Potentials (ErrP) have been used to include the perception of subjects on errors caused by the BCIs.

An ErrP originates from the anterior cingulate cortex and is elicited as a response to error perception after a feedback presentation [2]. It is characterized by a positive peak at around $200ms$ followed by a large negative peak at 200-$250ms$ and another positive peak at around $320ms$ [1].

Recent research on ErrP aims at designing new methods to detect this type of ERP. Such classification can improve BCI systems in two ways. Firstly, the system can automatically correct the erroneous action either by changing to the remaining action (in a binary set of choices) or by choosing the next most probable action intended by the user. Secondly, the system can use the error to improve its own prediction and perform better in a later instance of the same situation.

The current challenges in ErrP analysis include improving the detection accuracy, the generalization over different subjects [1] and computation time for real-time applications.

In this paper, we propose a new Convolutional Neural Network model with a reduced temporal input size and fewer layers compared to previous models and we test it using the Monitoring Error-Related Potential dataset [3]. We also discuss how generalization is not an issue in a personalized context where BCIs are tailored to each individual and propose future work to explore this idea.

[1]Instituto Superior Técnico, Lisbon
`jose.r.c.correia@tecnico.ulisboa.pt`
`jmrs@tecnico.ulisboa.pt`
[2]Politecnico di Milano, Milan
`luca.mainardi@polimi.it`

II. BACKGROUND

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, we focus on some recent work that implements CNN models and use them as a comparison to our proposed model.

In 2018, Torres *et al.* proposed a model called ConvNet [2] which was tested on two different input formats: one with only two EEG channels (*FCz* and *Cz*) and another with all available 64 channels. The latter reported better accuracy and is the one replicated here. It consists of a module (containing a mixed spatio-temporal convolution, an ELU activation function, and a max-pooling layer) repeated twice and a single fully connected layer with two output nodes. Before feeding the model with data, the input is randomly cropped to a size of $125ms$ to " reduces the probability of identifying a false training local minima."

Bellary *et al.* proposed, in 2019, a model using only two electrodes (*FCz* and *Cz*) called ConvArch [1]. In the first layer, a mixed spatio-temporal convolution without activation functions is followed by batch normalization, and dropout layers. Then, a module is repeated three times, each with a temporal convolution, ReLU activation function, batch normalization and dropout layers. In the end, a fully connected layer is used with two output nodes.

In 2018, Luo *et al.* proposed a model called CNN-L [4] where a major difference from the two previous models is the use of a sequential convolution (temporal followed by spatial) instead of a mixed one.

Due to space constraints, not all models are described here. For a full review of the compared models consult the first author's Github page [5], where description and schematics are shown for 7 different models.

III. METHODOLOGY

A. Dataset

The used dataset was created by Chavarriaga et al. [3] and is publicly available at the BNCI Horizon 2020 project website under the name *Monitoring error-related potentials*.

TABLE I

SUMMARY OF SOME ARCHITECTURAL ASPECTS FROM PREVIOUS LITERATURE TO BE CONSIDERED WHEN DESIGNING NEW MODELS. DASH (-) FOR NON-USED FEATURES AND QUESTION MARK (?) FOR USED FEATURES BUT NON-DEFINED VALUES.

	ConvArch (2019)	**ConvNet (2018)**	**CNN-L (2018)**
Input normalization	-	Pre-processing	-
Conv. order: Temporal (TC) vs Spatial (SC)	Mix	Mix	TC >SC
Activation function(s)	ReLU	ELU	ELU
Dropout rate	?	-	0.5
Number of fully connected layers	1	2	1
Number of output nodes	2	2	2

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 6 subjects (mean age 27.83 ± 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.

Concerning the dataset split for the training and evaluation of all the models, we simulated the real use of a BCI system, where the model is trained and only later tested with new data. Hence, the training is performed with the first session of the dataset while both the validation and testing sets use the second session. Balancing the dataset was achieved by replicating trials from the under-represented class.

B. *Pre-processing*

The only pre-processing applied to the raw data before being fed to the CNN model is a band-pass filter from 1 to $10Hz$. The input EEG data is organized in a 64×307 matrix where the number of rows, $C = 64$, is the number of EEG electrodes and the number of columns, $N = 307$, is the number of time samples (the sampling frequency is $512Hz$ and each epoch, starting from the feedback presentation, is $600ms$, leading to $N = 0.6 \times 512 \approx 307$).

The reason why all 64 channels are used and no electrode selection is done is two-fold. Firstly, according to our experiments, using all 64 channels consistently presented the best accuracy when compared with approaches that pre-select electrodes (two common electrodes for ErrP classification are the *FCz* and *Cz*). Secondly, as both the spatial resolution and the number of electrodes increases in EEG and ECoG imaging techniques [6] it is imperative that we investigate the best way to make deep learning models select or combine the best channels possible rather than hand-choosing specific electrodes which might lead to sub-optimal performance.

C. *Proposed model*

In this section, the proposed CNN model is detailed.

Firstly, we apply a batch normalization layer, using the same strategy as Liu *et al.* [7], where the normalization is applied directly to the raw data as it is reported to better control the overfitting effects that takes place during training.

Then, two convolution layers are used. First, a temporal one that extracts features directly from the normalized raw input and then a spatial convolution that combines temporal features from multiple electrodes. From all the previous models analyzed, only the CNN-L uses this same convolutional ordering, while others use either the reverse order or a mixed convolution. However, by performing the spatial convolution before the temporal or even mixing them, no purely temporal features are extracted because the temporal convolution is performed over abstract data. By performing temporal convolution before spatial convolution, we expect to better preserve and extract the raw temporal features. Also concerning the convolutions, previous models use unitary stride (the slide size during convolution) with large kernels which yields large amounts of overlap and repeating information. After testing with different sizes, we decided to use a stride size equal to the kernel size, which avoids overlap and largely decreases the complexity of the model.

After each convolution, a batch normalization is performed to center the data distribution before the ELU activation function. Bellary et al. does not apply an activation function after the first convolution [1] and this variation is also tested.

Finally, the fully connected (FC) stage is added. We determine the adequate number of layers by empirically testing with 1, 2, and 3 layers. Before each one of these FC layers, a dropout layer is applied with a constant dropout rate also to be tested with values of 0%, 20%, and 50%.

The output of the model is composed of a single node. Although most previous models use two output nodes, it suffices to use only one node together with a Binary Cross-Entropy (BCI) loss function. This avoids redundancy and decreases the number of trainable parameters in the fully connected layer by half.

No pooling layer is used as the convolution stride used already decreases the size of the data matrices significantly and in this way no information is lost by downsampling data.

Figure 1 shows a schematic overview of the generic model (upon which the referred variations can be tested).

D. *Epoch window*

An important consideration for using these models in real-time applications is the temporal epoching window size of

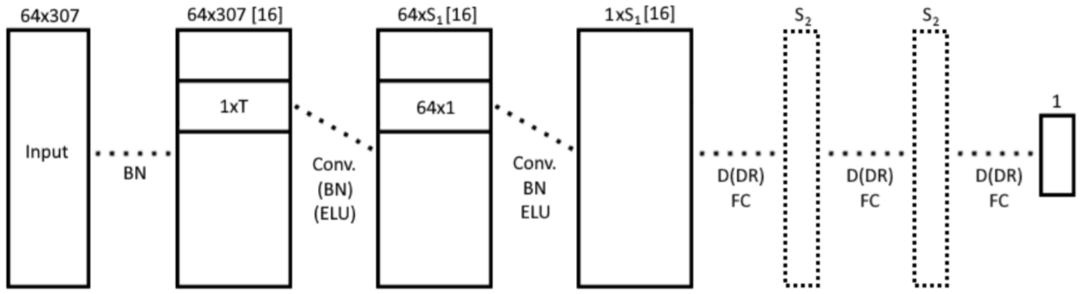

Fig. 1. Generic architecture for the proposed models. Data matrix dimensions on top (number of maps inside square brackets), kernel dimensions inside the rectangles. Variables refer to some of the aspects of the model to be tested. T is the kernel size of the temporal convolution. S_1 and S_2 refer to dimensions of the data matrices which depend on T. Testing the use of a different number of FC layers is represented with the dotted data matrices.

the input. A larger input may allow the model to search for more temporal 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 temporal information and minimizes the lag-time during online settings.

Most other models use around $1000ms$ or more of the input. This means that, after the feedback presentation, the model must wait 1 second before being fed with the EEG signal for processing which is a considerable amount of time.

In this work, we try to decrease this lag by testing shorter epoch windows. Maintaining $1000ms$ as the control input size, we test windows of $600ms$, $500ms$, $400ms$, $300ms$ and $200ms$. Although decreasing the input size produces the desired effects of decreasing the lag time of the processing pipeline, the performance of the model should not be compromised. Therefore, the performance resulted from each window size is compared in the next chapter.

E. Training

All models are codded using the PyTorch Lightning framework [8] and trained with GPU acceleration (Tesla P100-PCIE-16GB). The training parameters for the comparison models were kept the same as in their original papers or educatedly guessed when not presented, such as dropout rates (20% rate used), optimizers (SGD used), or loss functions (Cross-Entropy used). For ConvNet, we did not apply the cropping technique to the input described by Torres et al. as it kept the accuracy close to random.

For our model, we used a batch size of 128, a learning rate of 10^{-3}, and an SGD optimizer with momentum and weight decay of 0.9 and 10^{-5}, respectively. Early stopping was implemented to reduce the necessary training time.

All the models and code necessary to reproduce the experiments are present in the first author's Github page [5].

IV. Results and Discussion

All results reported are averaged over five training sessions, each taking an average of 3 minutes to complete. ANOVA and t-tests are used for statistical analysis with Bonferroni correction and $\alpha = 0.05$.

TABLE II
Accuracy of proposed model when training with different epoched temporal windows (best value per metric in bold)

Epoch time range (ms)	Overall		
	Accuracy	Sensitivity	Specificity
[0,1000]	79.1% ±0.5%	75.2% ±1.0%	82.9% ±1.0%
[0,600]	**79.4%** ±0.7%	**76.4%** ±2.0%	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 results for the temporal epoch window experiment are present in Table II. It is clear that reducing the epoch window size from $1000ms$ to $600ms$ or $500ms$ does not negatively affect the accuracy of the model (the accuracy difference to the control is not statistically significant: $p = 0.647$ and $p = 0.277$, respectively). The shorter epoch sizes present statistical difference on the accuracy, sensitivity and specificity (with p-values for the accuracy difference of 0.012, 0.001 and 0.000 for $400ms$, $300ms$ and $200ms$, respectively). Because the $600ms$ range preserves the performance of the model (with the highest accuracy and sensitivity) and cuts the lag in about half, it is the window used for our model.

Table III shows the results for the new proposed model and some of the models used for comparison.

We started by determining the best model amongst the previous ones. An ANOVA test verifies that there is significance difference between model accuracies ($p = 0.001$) and *post-hoc* t-tests between ConvArch and CNN-L ($p = 0.001$) and between ConvNet and CNN-L ($p = 0.016$), reveals that

TABLE III

REPLICATED RESULTS FOR THREE PREVIOUS CNN MODELS COMPARED TO OUR PROPOSED MODEL (BEST VALUE PER METRIC IN BOLD)

Model	Performance metrics		
	Accuracy	Sensitivity	Specificity
ConvArch	71.2% ±1.7%	56.1% ±6.7%	**85.4%** ±3.2%
ConvNet	76.0% ±0.8%	67.1% ±3.8%	76.3% ±0.8%
CNN-L	77.6% ±0.7%	71.7% ±2.8%	83.1% ±1.5%
Proposed model	**80.4%** ±0.7%	**75.9%** ±1.5%	84.7% ±1.5%

the best model is CNN-L.

Next, we determined our best proposed model out of the variations introduced in the *Proposed model* section by fixating all parameters and testing the accuracies produced by varying one single parameter at a time. For the best model, the temporal kernel size was set to $T = 20$ time samples and the first convolution layer was not followed by batch normalization nor an ELU activation function. One fully connected layer was used with a dropout rate of 20%.

Finally, we verify that, regarding accuracy, our model outperforms all previous models and, in particular, the best previous model (CNN-L) with statistical significance ($p = 0.0004$).

Concerning the replicated models from previous studies, both ConvNet and ConvArch models report higher accuracy while using the same dataset. However, for the purposes of the present work, only the model networks were recycled while all other procedures such as pre-processing, dataset balancing, and dataset split were uniformized for a fair comparison. From the results published by Bellary et al., it appears that their dataset contains a non-uniform distribution of subjects, showing an over-representation of subject 1. Since this particular subject presents one of the highest accuracies, it introduces a bias in the final reported accuracy.

The random cropping technique used in ConvNet is based on suggestions from previous work [9] to effectively increase the dataset size. However, in [9] the percentage of cropped data compared to the total size is much bigger than that used by Torres et al. ($< 10\%$). Although being a good technique for data augmentation, randomly partitioning the data into such small chunks slows and even prevents the learning process of the network and that is why we chose not to apply it in this study.

Finally, we address the commonly cited problem of generalization. We believe this only constitutes an issue in a *one-size-fits-all* paradigm where BCI models are trained by the data from several subjects. However, in many current applications, BCIs are used in very personal contexts such as allowing disabled or locked-in patients to communicate or control a wheel-chair [10]. Hence, a more tailored approach must be taken where models can learn generic feature detection from a large population but also be able to learn the best correlation of such features for individual subjects.

V. CONCLUSION

In this work, a new model for the classification of error perception is proposed, which outperforms previous models. Our shallower architecture together with regularization methods such as the batch normalization and dropout layers provides an effective model to classify the presence of error-related potentials and hence the error perception of a user.

The temporal epoch window is reduced from the typical $1000ms$ to $600ms$ while maintaining the same level of accuracy, which effectively decreases the lag between acquisition and processing in real-time applications.

As future work concerning the generalization problem mentioned previously, we suggest using transfer learning to achieve the proposed solution regarding generalization: first, a CNN model is pre-trained on a dataset with a large group of 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.

REFERENCES

[1] 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)*. jan 2019, pp. 245–249, IEEE.

[2] 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)*. jul 2018, vol. 2018-July, pp. 360–363, IEEE.

[3] 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*, vol. 18, no. 4, pp. 381–388, aug 2010.

[4] 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)*. dec 2018, pp. 697–701, IEEE.

[5] Rafael Correia, “Error perception classification in bci using cnn,” https://github.com/LeafarCoder/Error-perception-classification-in-BCI-using-CNN, 2020.

[6] Elon Musk, “An Integrated Brain-Machine Interface Platform With Thousands of Channels,” *Journal of Medical Internet Research*, vol. 21, no. 10, oct 2019.

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

[8] WA Falcon, “Pytorch lightning,” *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, vol. 3, 2019.

[9] 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*, vol. 38, no. 11, pp. 5391–5420, nov 2017.

[10] Ujwal Chaudhary, Niels Birbaumer, and Ander Ramos-Murguialday, “Brain–computer interfaces for communication and rehabilitation,” *Nature Reviews Neurology*, vol. 12, no. 9, pp. 513–525, sep 2016.