Decoding P300 Waveform plots based on Convolutional Neural Networks

1st Brian Ezequiel Ail
Computer Engineering Department
Instituto Tecnológico de Buenos Aires
Buenos Aires, Argentina
bail@itba.edu.ar

2nd Rodrigo Ramele
Computer Engineering Department
Instituto Tecnológico de Buenos Aires
Buenos Aires, Argentina
https://orcid.org/0000-0001-8155-0124

3rd Juan Miguel Santos
Computer Engineering Department
Instituto Tecnológico de Buenos Aires
Buenos Aires, Argentina
jsantos@itba.edu.ar

***Abstract*—The use of Brain-Computer Interfaces can provide substantial improvements to the quality of life of patients with diseases such as severe Amyotrophic Lateral Sclerosis that could potentially derive in Locked-In syndrome, by creating new avenues in which these people can communicate and interact with the outside world. The P300 speller is an interface which provide the patients the ability to spell letters and eventually words, so that they can speak while unable to use their mouth. The P300 speller works by reading signals from the brain using an Electroencephalogram. Traditionally, these signals were plotted and interpreted by specialized technicians or neurologists, but the development of Machine learning algorithms for classification allow the computers to perform this analysis and detect the P300 signals, which is an Event Related Potential triggered when certain stimuli such as a bright light is triggered on a place that the patient is focused on. In this thesis we used a Convolutional Neural Network to train multi-channel EEG readings, and attempted to detect P300 signals from a P300 speller. The results are corroborated against a public ALS dataset.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

The understanding of the human brain has been one of the most exciting fields in recent history. The term psychokinesis and telekinesis has been around in science fiction for more than a century, from Luke Skywalker recalling his lightsaber using the Force, to Magneto or Jean Grey of the X-Men. And what is telekinesis if not the movement of a physical object through signals emanating from the human brain? [1] The technology is not quite there at the moment, but it does not seem too far fetched to imagine right now, and not at all in the realm of science fiction. Controlling a prosthetic limb with the mind, or a monkey playing Pong [2] with just his brain, are things that would have been unimaginable only a few years ago, but cutting edge companies such as Neuralink are already working on invasive "high-bandwidth brain-machine interface system" [3], which can make this kind of things a reality [4].

In terms of potential medical applications, patients with severe cases of Amyotrophic lateral sclerosis (ALS) can get into a 'Locked-in' state, in which they are unable to move any of their muscles to communicate. Creating a Brain-Computer interface my be the only remaining way to allow them to connect with the outside world and substantially improve their quality of life [5]. This has been the main goal of this discipline.

The aim of this work is to analyze the P300 signals in terms of Actionable Analytics and emphasizing interpretability. Deep learning black box problem which is solved by this approach. Feature engineering can be useful in this situation. Because after all DL is still not very usable in this situation [6].

This work proceeds as follows. Section **??** introduces several approaches to decode EEG signals in this rich field. Then we will discuss about our proposal for classification, which involves plotting the signals, and training a Convolutional Neural Network to identify the images. The Materials and Methods explains the control procedure implemented to verify the theoretical validaty of the proposed algorithm. Finally, the Results section compare these results to other studies performed using a public and reliable dataset of EEG signals from ALS patients.

A. Background

As mentioned in section 1, the advance of hardware technologies, as well as the cheap accessibility of portable EEGs have significantly advanced the study of EEG signals using DNN. According to a 10 year study of Deep Learning in EEG [7], the amount of published papers that matched the searching criterion n ["Deep Learning" AND "EEG" AND "Classification" OR "Recognition" OR "Identification"] after manually curating the list, 213 papers had been published by march 2020. Out of these the vast majority of them were published from 2019 to 2020, and the trend is increasing rapidly. This papers include all range of studies of EEG, varying from Brain-Computer Interfaces to other applications such as Disease Detection, Sleep stage classification and Emotion recognition.

When discussing applications of Deep Networks in BCI, they were a little hesitant, as Convolutional Networks are often used for computer vision or speech recognition, and did not match the requirements needed to analyze EEG signals. However they found many papers of successful applications of Convolutional Networks such as the work of Li et al. [8], which uses 3 different blocks of convolutional networks to decode motor imagery, which if successful would allow patients with loss of motor function to recover this lost mobility by the use of external devices. Liu et al. [9] used Batch Normalization to speed up the training and alleviate the

overfitting. There are many other papers describing not only Deep Architectures for Brain Computer interfaces, but also for disease detection, mainly focusing on Epilepsy (with almost 50% of the published papers) and Depression in second place with about 10%.

Another review by Roy et al. [10] specifically about using Deep Learning on EEG signals review a total of 156 papers. First of all, there is a very similar graph showing an increasing trend in the amount of published papers over the years, rising from less than 10 in 2014 to over 50 in 2018.

It also goes over most of the common problems, which we have also run into in this work. First of all regarding data imbalance, most papers with imbalanced data, such as epileptic seizure detection, or sleep stage transitions, used some form of class balancing, either by subsampling the majority class or by resampling the minority class on training.

For the DL architecture, the majority of the works used convolutional neural networks, with an overwhelming 41%, while other architecture types such as autoencoders (AE) or recurrent neural networks (RNN) are under 14%.

Regarding the amount of layers used, most architectures used from 2 to 7 layers, with few examples over 7 layers,

Regarding regularization, 76 papers used some form of regularization, such as L1 or L2, dropout, or early stopping. While the other 80 do not mention regularization.

Finally, most papers (30%) used the Adam method for optimization, while 17% used Stochastic Gradient Descent, and 6% used different optimizers.

This review gives us two big takeaways, first of all, it validates most of the choices used in the architecture proposed in our research. Another takeaway from both reviews, is that while Deep Learning seems like a big promise for the field of EEG, the field is still in it's early steps, the Deep Networks do not perform nowhere as good on EEG signals as they perform on other applications such as computer vision or speech recognition. So it is still too early to say if Deep networks can perform as good as other learning algorithms, but if they can reach the same accuracy as they do on other applications, it can be a practical tool for EEG analysis.

II. Materials and Methods

A. Software and Hardware

The code ran on a HP Pavillion laptop with a Intel I7 @2.8GHz processor. The available RAM was 16Gb, although the software could run on much less RAM. A GPU was available, but the software did not run on a GPU.

The software for the signal segmentation, processing, and Signal drugging was written in python, using a modified version of the EEGWave repository from codeocean [11]. It uses the MNE library for segmenting the data and some operations with numpy for signal processing. The signal plotting is written in C++ using OpenCV, And the NN was run using Tensorflow's API for C++, based on Benny Friedman's article [12]. The full code is public and can be found at https://github.com/shipupi/BciSift/.

B. P300 Experiment

1) Experiment Description: The experiment was performed by Riccio et al., 2013 [13]. where a group of eight individuals with confirmed ALS disease were tasked to spell 7 5-letter words (35 total letters) with a P300 speller.

The first three words were used for calibration/training, while the remaining four were used for testing with visual feedback.

Each time a letter was attempted to be spelled is called a Trial. Each trial contained 10 flashes in each of the 6 rows, and 10 flashes in each of the 6 columns. The flashes lasted a total of 0.125 seconds each, followed by an inter-flashing pause of the same duration. After each 120 flashes, an inter-trial pause was performed before moving to the next letter.

The experiment was performed with an 8 channel EEG, with electrodes placed at the Fz, Cz, Pz, Oz, P3, P4, PO7, and PO8 channels according to the 10/20 international system. The software used for the flashing was the BCI2000 (open source) [14]

The subjects were instructed to perform a copy-spelling task, which means that they have to spell a predetermined set of words that was instructed beforehand.

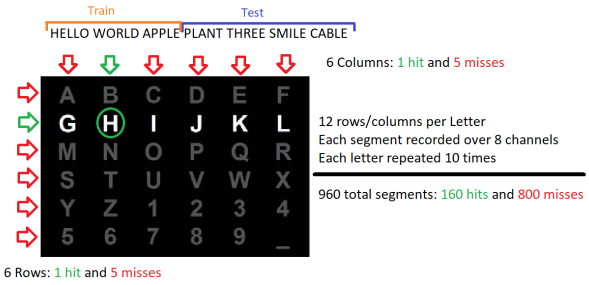


Fig. 1: For each of the 35 letters, each row and column is highlighted 10 times, to allow for signal averaging. In each of these stimulus, if the column or row contains the target letter, we consider it a 'hit', while the columns and rows without the target letter are considered misses or 'nohits'. Since our EEG reads a total of 8 channels, the amount of total segments is calculated as 12 rows/columns x 10 repetitions x 8 channels giving a total of 960 segments per letter. Since only a row and a column contain a hit while the others contain misses, we have 160 segments with hits and 800 segments with nohits.

2) Dataset Structure: After processing and segmenting the raw data. We can calculate the amount of segments generated by a single letter as follows:

$$N_l = (n_r + n_c)rc \tag{1}$$

Where n_r and n_c are the number of rows and columns in the P300 matrix, c is the amount of channels i.e. the amount of electrodes on the EEG, and r is the amount of repetitions done per letter.

In the ALS experiment, the data was recorded using an 8-channel EEG, and 10 repetitions were performed per letter. The P300 matrix contained 6 rows and 6 columns, this results

in a total of 960 segments per single letter. There is a single row and a single column with the P300 signal, and 5 rows and 5 columns without it. This results in each letter being segmented into 160 hits, and 800 nohits.

The experiment was performed using 7 5-letter words, resulting in 35 letters, and a total of 33600 segments. Out of these, 15 letters were used for the training of the networks, and the remaining 20 were used to test the accuracy.

Explain the three networks.

C. VGG16 Neural Network

The first version of the neural network was based on VGG16, a deep network which was used to win the ImageNet Large Scale Visual Recognition Challenge (ILSVR) competition in 2014 [15]. It uses a 3x3 filter with a stride of 1 and uses padding to keep the same spatial dimensions, followed by a MaxPool layer of 2x2 size and stride 2, and follows this arrangement of convolution and maxpool layers all the way throughout the whole architecture, for 5 convolutional + maxpool sets. In the end, the VGG16 has 2 fully connected (regular neural network) layers, while we are using 4. In the end, the last layer is activated with a sigmoid function to make the binary classification. The advantage of the VGG16 approach, is that it reduces the amount of hyperparameters that must be tinkered with, as all the filters and strides are kept constant.

Our network has 6 convolutional layers with a depth of 1, 32, 64, 128, 128 and 256. The stride in the maxpool reduces the spatial size in each iteration, leaving a spatial size of 150, 75, 38, 19, 10, 5 (the images are squared, so width and height are the same).

After the convolutional layers, a dropout layer with a drop rate of 0.5, and a flatten layer to make the data ready for the dense layers. The 4 dense layers have a size of 6400, 1024, 512 and 256 units, with the last one having a sigmoid activation.

The learning algorithm was the stochastic gradient descent method called Adam [16], and the loss function was the means squared difference. And it was done in batches of size 20.

The learning rate used was $5\,10^{-4}$, which deviates from the recommended $3\,10^{-3}$. This parameter was tinkered with for a bit, and was the suspect of the Network not convering at first, so decreasing the learning rate helped a bit.

Ultimately, the convergence issue was solved by balancing the dataset, first by prunning some of the misses in, and in later versions by duplicating the hits, as this latter method allowed the use of all the misses.

After training with the first 15 letters, the network was used to predict the remaining 20 letters, and the performance of the network was calculated by finding the row and column with the highest predicted change of containing the P300 signal and comparing the predicted letter to the original expected letter from the experiment.

These predictions were done by training each of the EEG channels separately, and then finding the channel that best perform. As mentioned early, another prediction was used by making a non-weighted voting system by each of the channels, and selecting the row and column with the most votes.

Since we are running what is considered a 'BCI Simulation'. The training was performed only once, instead of multiple runs doing cross validation or calculating mean and standard deviation in the accuracies. This is because we want to simulate a real use case of a patient actually using our interface.

D. Small VGG16

The second approach was the Small VGG16 (SV16). The first change was to reduce the network size, so 2 convolutional layers and 2 fully connected layers were removed. Leaving the convolutional layers depth at 1, 32, 64 and 128, with spatial sides of 150, 75, 38, and 19. The flatten and dropout layers are left untouched, and then finally 3 fully connected layers of sizes 46208, 512 and 256. The overall philosophy of the architecture was left untouched, i.e. the padding, filter size and stride.

Another addition in this version was early stopping at 7 epochs to help decrease overfitting.

E. Multichannel Small VGG16 (MSV16)

With this multichannel network, instead of training all of the EEG channels separately, they were merged into a single 8-channel image on the preprocessing stage. And the network was trained with this image as input. This modified the input layer to have a depth of 8, but the subsequent layers were left intact. The best improvement was that it allowed for the combination of all the different EEG channels to work together, achieving a higher performance than each channel separately.

To make this change work however, extra changes had to be made to the network parameters. The batch size was reduced to 6, as having a greater batch size generated memory issues. The last change was reducing the learning rate to the recommended value of $3 \cdot 10^{-}3$.

III. Results

A. Validation on the Pseudoreal dataset

Subject	VGG16(%)	SV16 (%)	MSV16(%)	HIST(%)
1	15	10	0	35
2	70	50	75	85
3	30	30	40	25
4	30	40	30	55
5	35	50	50	40
6	45	40	50	60
7	70	65	80	80
8	90	95	100	95

TABLE I: Character recognition rates for VGG16, SV16, MSV16 and HIST

The third and final version of the network included both the improvements made in SV16, as well as the addition of multi-channel classification, and a smaller batch size. The results can be seen in I. This last version showed an accuracy increase in

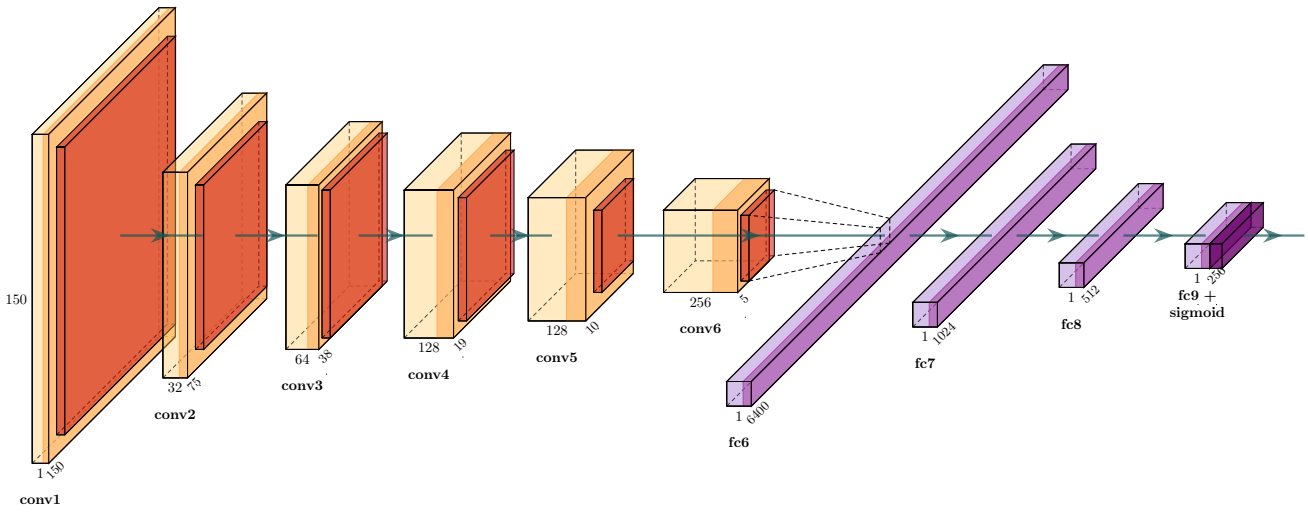


Fig. 2: First version of the NN, a set of 5 convolutional layers is followed by 4 fully connected layer, and finally activated using a sigmoid function

6 out of 8 channels over the other two versions. It surpasses HIST in 3 out of 8 subjects, and performs equally on 1. It does however seem to underperform on subject 1 and 4 compared to earlier versions and HIST. But it also reaches a 100% accuracy on subject 8, which none of the other methods had managed to achieve. Also, by looking at the accuracy per intensification level, subject 8 achieved over 90% accuracy in only 5 repetitions, meaning that a robust speller could be established with a much faster transmission rate.

Classification results are also shown from [17].

The learning curve for subject 5 at an intensification level of 4 can be seen in 6. If compared to the same curve in the first network **??**, the accuracy for the validation set is much higher, while the accuracy of the training set remains at or close to 100%. This suggests that there is still some overfitting left to solve, which would indicate that this method still has potential to find even higher success.

IV. Discussion

1) Results validation: While there is no fail-proof way of assuring that the code is running bug-free, certain automatic and manual fail safes have been included to make the assertion that the results are valid.

When generating the training data, only the 15 letters are plotted. The remaining 20 letters are completely ignored, there is no possibility that the training used any of the testing set.

Generating the testing set follows a similar process, as the plotting is only performed on the last 20 letters, and the plots for the first 15 letters are deleted from the hard drive.

When averaging the signals, the standard deviation of the resulting signals is calculated, and it is asserted that the new standard deviation is indeed lower than the original one.

To corroborate that the results were not random, a training of the dataset was performed by randomizing the labels, the expected outcome of this is that the predictions for the letters are completely random. We can see this in figure 7, the accuracy of the training done with random labels on subject 8 with MSV16 is oscillating around the random threshold, which is 1/36, or 3%, while the same training without randomizing the labels on the training goes back up to the regular accuracies. This shows that there is indeed a generalization being performed by the network, and the results are not 'by chance'.

The most telling assurance however, is that the results are consistent with previous works on the same dataset such as [18]. The accuracy curves for the different subjects, while not exactly identical, follow similar patterns, such as the subject 8 performing very well, while subject 1 performs badly. And the accuracy is consistently increasing along with the intensification level.

2) Drugged signals: When we generate a drugged dataset with a boost level of 0, we get a purely basal EEG signal, when encountering this dataset, the network should learn and converge, but the accuracy on the testing set should perform very bad, close to random levels, as there is no pattern to generalize. This can be seen on figure **??**, while the training accuracy (red line) increases, the testing accuracy (yellow line) is stuck around 50%, which means that the prediction is completely random.

When moving on to a boost level of 1, the signal is very similar to the subject 8 of the ALS dataset, this gave us a preview of how the network behaved and learned on a real dataset. On figure **??** the boosted 1 curve (red line) performs pretty well, reaching almost 50% accuracy when averaging

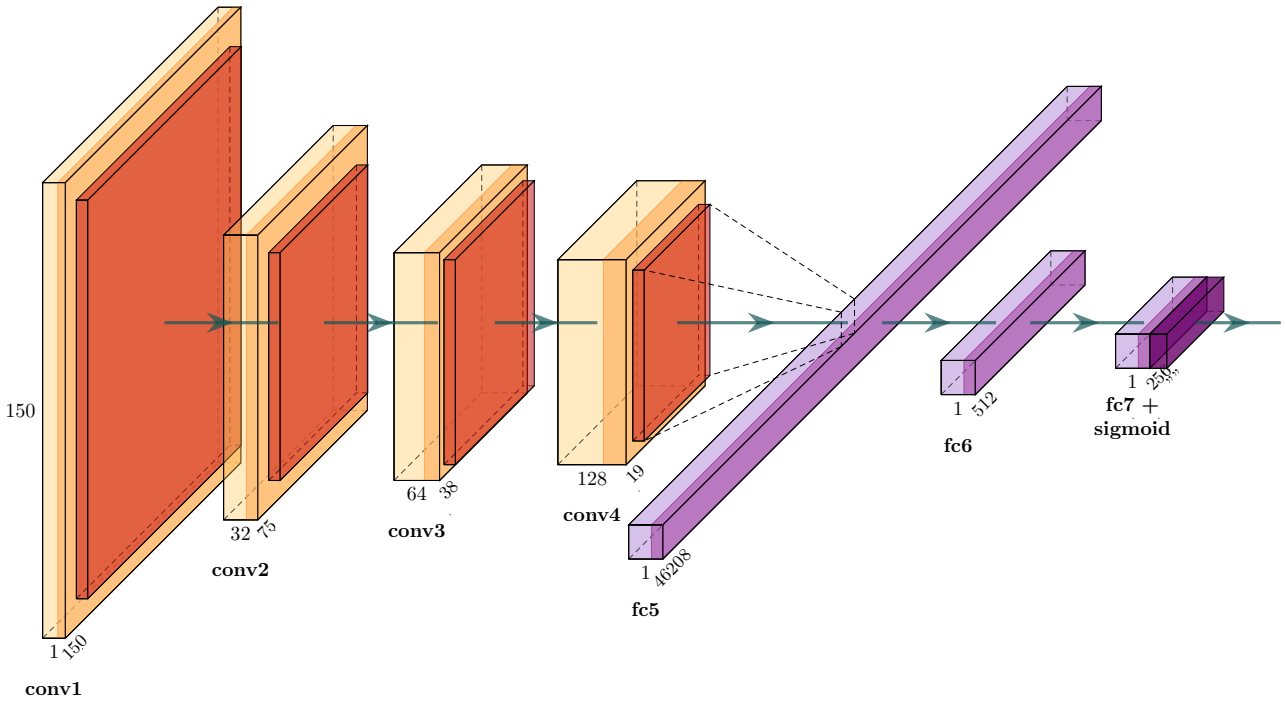


Fig. 3: SV16 is similar to VGG16, but has two less convolutional layers and two less dense layers

10 signals (note: the threshold for random letter accuracy is 1/36≈3%).

As the boost levels increased, the P300 became more dominant in the signal and the SNR decreased, making the images easier to classify, thus the expectation is that when the level of boost increases, the accuracy increases as well, which we can see on figure **??**.

3) DL performance: Regarding the VGG16 architecture, we can see in figure **??** that the main issue is overfitting. The training sets are being learned perfectly, reaching a 100% accuracy, but the accuracy on the validation set does not increase, and oscillates around 60%, while this is a bit better than random level values, it still needs improvement. Figure **??** is just one of the many learning curves drawn for this version, but it showcases the overfitting problem.

The SV16 version attempted to address this issue by adding early stopping and reducing the amount of layers, as shown on table **??**, the improvement on letter accuracy from these changes seems small, as the results are barely better, but it should be taken into account that the results for VGG16 are taken with the best performing channel, while the results for SV16 are only taken from training on the PO8 channel, so even a similar result is a good indicator.

The biggest breakthrough of this work however is the MSV16 version, the inclusion of all the EEG channels into a single image, generated substantial improvements compared to both previous versions, and it even surpassed other proven methods such as SVM and HIST on some of the subjects, proving that using DL methods for this task is a very viable option. For some reason however, it performed very poorly on subject 1, although most of the DL methods did, while the HIST method can get a 35% accuracy on it.

ACKNOWLEDGMENT

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks . . .”. Instead, try “R. B. G. thanks. . .”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [**?**]. The sentence punctuation follows the bracket [**?**]. Refer simply to the reference number, as in [**?**]—do not use “Ref. [**?**]” or “reference [**?**]” except at the beginning of a sentence: “Reference [**?**] was the first . . .”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be

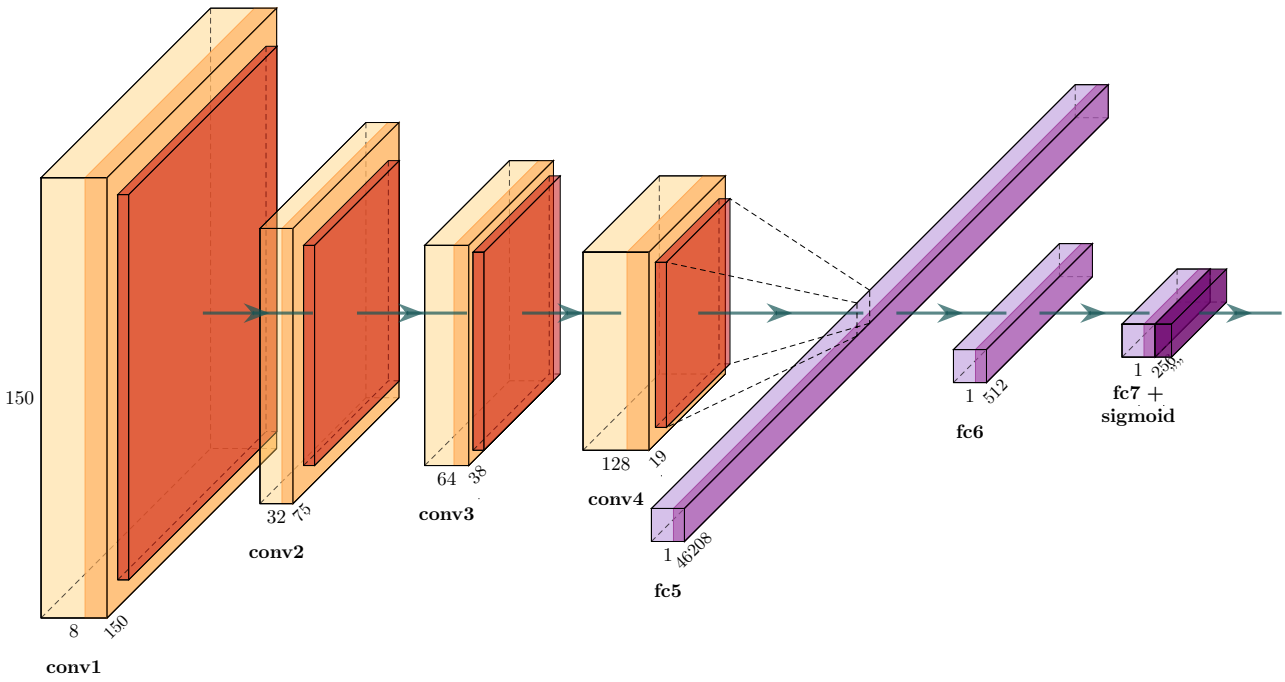


Fig. 4: MSV16 has the same architecture as the second one, but the input layer is modified for an 8-channel input

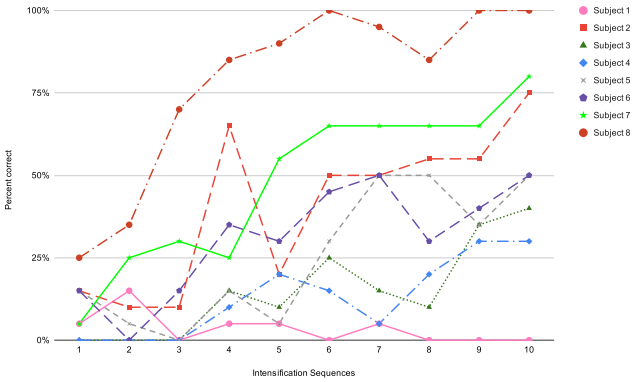


Fig. 5: Letter accuracy for MSV16 shows a significant improvement over SV16

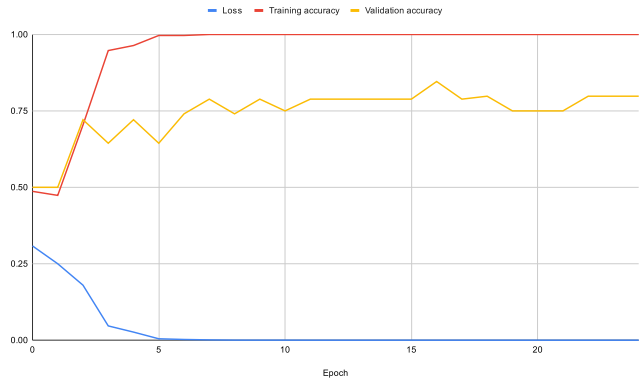


Fig. 6: Learning curve of subject 5, with an intensification level of 4 using multichannel training

cited as "unpublished" [**?**]. Papers that have been accepted for publication should be cited as "in press" [**?**]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [**?**].

REFERENCES

[1] S. Bozinovski and L. Bozinovska, "Brain–computer interface in europe: the thirtieth anniversary," *Automatika*, vol. 60, no. 1, pp. 36–47, 2019. [Online]. Available: https://doi.org/10.1080/00051144.2019.1570644

[2] Neuralink. (2021) Monkey MindPong . [Online]. Available: https://neuralink.com/blog/monkey-mindpong/

[3] E. Musk, "An integrated brain-machine interface platform with thousands of channels," *J Med Internet Res*, vol. 21, no. 10, p. e16194, 10 2019. [Online]. Available: http://www.jmir.org/2019/10/e16194/

[4] J. E. Huggins, D. Krusienski, M. J. Vansteensel, D. Valeriani, A. Thelen, S. Stavisky, J. J. Norton, A. Nijholt, G. Müller-Putz, N. Kosmyna, L. Korczowski, C. Kapeller, C. Herff, S. Halder, C. Guger, M. Grosse-

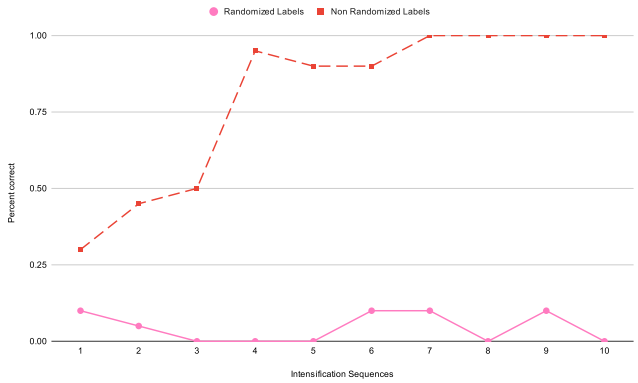


Fig. 7: Accuracy of MSV16 on subject 8, the red line shows a normal prediction using regular labels on the training set, while the pink line shows the predictions of the NN by training it with randomized labels on the training set. We can see that the accuracy on the pink line hovers around the *random* range(3%)

Wentrup, R. Gaunt, A. N. Dusang, P. Clisson, R. Chavarriaga, C. W. Anderson, B. Allison, T. Aksenova, and E. Aarnoutse, "Workshops of the eighth international brain–computer interface meeting: Bcis: the next frontier," *Brain-Computer Interfaces*, vol. 9, no. 2, pp. 69–101, 2022. [Online]. Available: https://doi.org/10.1080/2326263X.2021.2009654

[5] V. Guy, M.-H. Soriani, M. Bruno, T. Papadopoulo, C. Desnuelle, and M. Clerc, "Brain computer interface with the p300 speller: Usability for disabled people with amyotrophic lateral sclerosis," *Annals of Physical and Rehabilitation Medicine*, vol. 61, no. 1, pp. 5–11, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877065717304104

[6] A. Kawala-Sterniuk, N. Browarska, A. Al-Bakri, M. Pelc, J. Zygarlicki, M. Sidikova, R. Martinek, and E. J. Gorzelanczyk, "Summary of over fifty years with brain-computer interfaces—a review," *Brain Sciences*, vol. 11, no. 1, 2021. [Online]. Available: https://www.mdpi.com/2076-3425/11/1/43

[7] S. Gong, K. Xing, A. Cichocki, and J. Li, "Deep learning in eeg: Advance of the last ten-year critical period," 11 2020.

[8] Y. Li, X.-R. Zhang, B. Zhang, M.-Y. Lei, W.-G. Cui, and Y.-Z. Guo, "A channel-projection mixed-scale convolutional neural network for motor imagery eeg decoding," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 6, pp. 1170–1180, 2019.

[9] M. Liu, W. Wu, Z. Gu, Z. Yu, F. Qi, and Y. Li, "Deep learning based on batch normalization for p300 signal detection," *Neurocomputing*, vol. 275, pp. 288–297, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217314601

[10] Y. Roy, H. Banville, I. M. Carneiro de Albuquerque, A. Gramfort, T. Falk, and J. Faubert, "Deep learning-based electroencephalography analysis: A systematic review," *Journal of Neural Engineering*, vol. 16, 05 2019.

[11] R. Ramele, A. J. Villar, and J. M. Santos, "Eeg waveform analysis of p300 erp with applications to brain computer interfaces," *Brain Sciences*, vol. 8, 2018.

[12] B. Friedman. (2019) Creating a tensorflow cnn in c++. [Online]. Available: https://towardsdatascience.com/creating-a-tensorflow-cnn-in-c-part-2-eea0de9dcada

[13] A. Riccio, L. Simione, F. Schettini, A. Pizzimenti, M. Inghilleri, M. Olivetti Belardinelli, D. Mattia, and F. Cincotti, "Attention and p300-based bci performance in people with amyotrophic lateral sclerosis," *Frontiers in Human Neuroscience*, vol. 7, 2013. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnhum.2013.00732

[14] G. Schalk, D. Mcfarland, T. Hinterberger, N. Birbaumer, and J. Wolpaw, "Bci2000: a general-purpose brain-computer interface (bci) system," *IEEE Trans. Biomed. Eng.*, vol. 51, pp. 1034–, 07 2004.

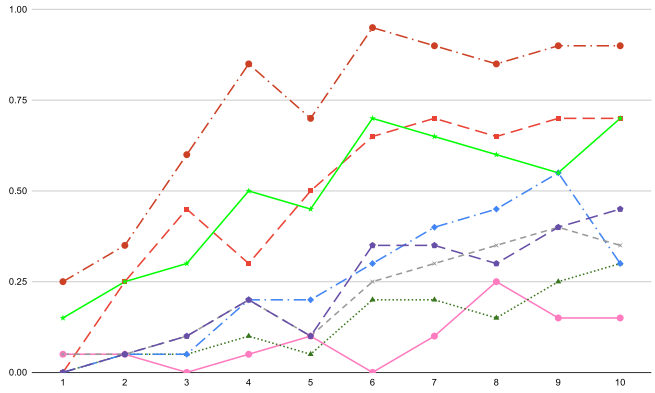
[15] "Imagenet large scale visual recognition challenge 2014 (ilsvrc2014)," https://www.image-net.org/challenges/LSVRC/2014/.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

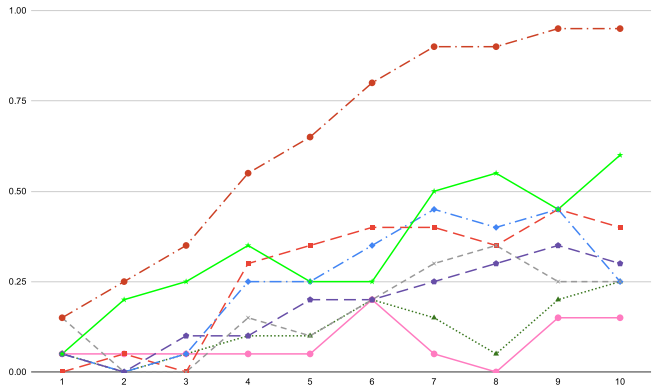
[17] X. Zhang, D. Wu, L. Ding, H. Luo, C.-T. Lin, T.-P. Jung, and R. Chavarriaga, "Tiny noise, big mistakes: adversarial perturbations induce errors in brain–computer interface spellers," *National Science Review*, vol. 8, no. 4, 09 2020, nwaa233. [Online]. Available: https://doi.org/10.1093/nsr/nwaa233

[18] R. Ramele, A. J. Villar, and J. M. Santos, "Histogram of gradient orientations of signal plots applied to p300 detection," *Frontiers in computational neuroscience*, vol. 13, 2019.

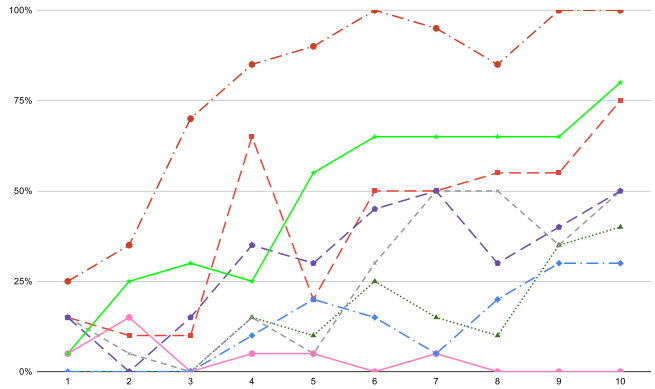
IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.



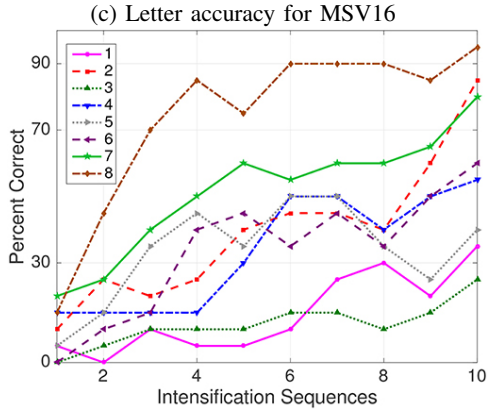
(a) Letter accuracy for VGG16



(b) Letter accuracy for SV16



(c) Letter accuracy for MSV16



(d) Letter accuracy for HIST method [18]

Fig. 8: Side by side comparison of all the architectures, and finally a comparison to the HIST method, original figures can be found on section **??**