Sleep EEG Signal Processing Guidelines

1st Aylin Vazquez Chenlo
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.**

This work proposes a method to experiment on EEG signals. This work proposes a method to create actionable explainable analytics on EEG waveform components.

***Index Terms*—BCI,EEG,P300,Waveform,CNN,Pseudoreal**

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? [**?**]. Recent advances have shown impressive achievements like controlling a prosthetic limb with the mind [**?**] or a monkey playing the computer game Pong [**?**], [**?**]. Although these impressive developments are based on invasive electrodes, a steady improvement albeit with modest results, is also being attained with noninvasive approaches [**?**].

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, and more traditional Alternative Augmentation Communication (AAC) devices [**?**], that harness information from any remaining muscle activity, are not effective. Creating a Brain-Computer Interface (BCI) may be the only remaining way to allow them to connect with the outside world and help them to improve their quality of life [**?**], [**?**].

The P300 is a positive deflection that appears on EEG signals around 300 ms after the onset of an unexpected stimulus that provides relevant information. It can be cleverly used to implement a speller device that exploits the appearance of this waveform indicating which particularly event a person is paying attention to. BCI systems based on P300 need to identify this voltage variation out of the natural noise of the EEG signal, and by doing that they can decode the message a person is trying to convey [**?**], [**?**].

CHEKCK The aim of this work is to analyze P300 signals emphasizing interpretability, by detecting automatically this component from the signal plot which is something that a clinician or physician can easily corroborate [**?**]. To do so, this work proposes to use Deep Learning (DL) techniques, particularly Convolutional Neural Networks (CNN) which have been proved very efficient to extract information from images, in this case images of signal plots. At the same time, this proposal tackles one of the most important issues of applying DL techniques to health information, which is the the black box nature of neural network architectures. This work empahsizes the usage of this form of Feature engineering in terms of Actionable Analytics [**?**], insight data that can be extracted from the problem, and that has a certain meaning in the knowledge base and dictionary used for the community that is currently dealing with the problem and at the same time using DL in a way which emphasize its real utility in this particular problem where data is scarce and DL is still not very usable in this situation [**?**].

This work proceeds as follows. Section I-A 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

Plethora of methods have been applied to decode EEG signals [**?**], [**?**] and particularly to decode P300 signals [**?**].

However only a few have focused on the idea of determining the presence of the P300 ERP response based on the waveform that can be obtained from a signal plot [**?**].

The success of Deep Learning in several fields [**?**], has opened the posibility of using this same technique to try to identify the complex patterns that can be detected from EEG signals. According to [**?**], the amount of published papers that matched the searching criterion [”Deep Learning” AND ”EEG” AND ”Classification” OR ”Recognition” OR ”Identification”] after manually curating the list, reflects that around 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.

Predicting EEG signal channels based on the ifnromation from other channels [**?**].

BCINet a convolutional neural network to predict population features from EEG signals and workload [**?**].

CNN to identify Motor Imagery patterns which are also used in BCI applications. [**?**], or Sleep Stage identification [**?**].

EEGNet [**?**] is a convolutional neural network which consists of 3 convolutional layers and 1 fully-connected layer, designed to process EEG data, using the convolution as a way to implement multiscle filter-banks [**?**].

xDAWN Variational Mode Decomposition VMD

A complete review regarding the usage of Deep Learning methods to identify EEG signals is outside of the scope of this work, and may deserve an independent review by itself. At the same time, the utility of DL in processing EEG signals, and health care information in general is still debatable, due to the issue of not being able to obtain enough information from each patient and the limitations of transfer learning approaches for health data. We circumvent that problem here because we are using DL-CNN for what we know they do well: decoding information from images.

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. [**?**], 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. [**?**] 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. [**?**] 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.

Authors in [**?**] presented an approach where CNN are used to interpret the information from scalogram images from a single channel electrode.

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. P300 Experiment

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

Each time a letter is attempted to be spelled is called a letter spell attempt [**?**]. Each letter spell attempt is composed of 10 flashes of the 6 rows, and 10 flashes of the 6 columns of the P300 matrix. Each flash lasts for a total of 0.125 seconds each, following by an inter-flashing pause of the same duration. After each 120 flashes, an inter-trial pause is performed before moving to the next letter.

The original experiment was performed with an 8 channel EEG (g.Mobilab, g.Tec, Austria), with a sampling frequency of 256 Hz 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 implementing the speller and the processing was the BCI2000 [**?**]. The eight subjects were instructed to perform a copy-spelling task, which means that they had to spell a predetermined set of words that was instructed beforehand. In the original experiment, and the approach used in this work, is to use the first three words

for calibration/training, while the remaining four are used for testing with visual feedback.

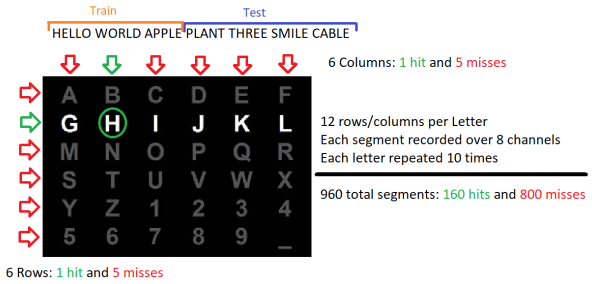


Fig. 1: For each of the 35 letters, each row and column is highlighted 10 times, to allow for signal averaging. In each of these stimulus, if the column or row contains the target letter, we consider it a 'hit', while the columns and rows without the target letter are considered misses or 'nohits'. Since 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.

As there are 10 repetitions per letter, 6 rows and 6 columns, this generates 120 flashes per single letter. Summing up on all the 8 channels, this results in a total of 960 segments, which are separated 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 in the BCI simulation.

B. BCI Simulation

The task of decoding information from brain signals inherits practices from Machine Learning (ML). Cross-validation is used in ML to reduce overfitting bias and to increase the independence on the dataset that is used as calibration. However, the brain data used in BCI is extracted from a person who is performing a task and whose signals are changing while trying to adapt to this operation. Hence, mixing the dataset, shuffling the sessions and trials is at least a challenging assumption. BCI Simulation, on the other hand, is not very well defined in BCI research, but their practice, without naming it, has been the regular approach for BCI Competitions. It consists in reproducing the operational sequence that was utilized to generate the dataset. Hence, the experiment is replicated offline using the training information to train or calibrate a classifier, and to classify the testing signals as if they were generated at that same moment [**?**]. 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.

C. Signal Preprocessing and Plot Generation

D. Neural Network Architectures

This work proposes and experiments with three network architectures to identify the pattern of the P300 on the signal plot.

1) 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 [**?**]. 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 plus MaxPool sets. The original VGG16 has 2 fully connected layers, while for this particular case we are using 4. In the end, the last layer is activated with a sigmoid function to make the binary classification.

The network implementation 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 $150, 75, 38, 19, 10, 5$ (image plots 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 [**?**], 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.

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.

2) 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,

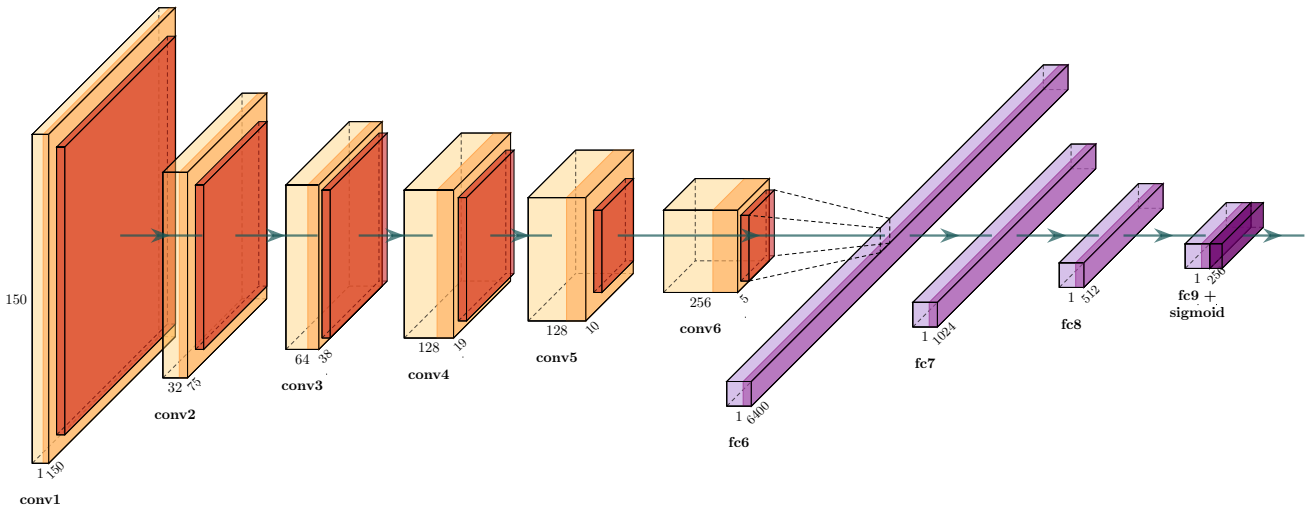


Fig. 2: First version of the NN, a set of 5 convolutional layers is followed by 4 fully connected layer, and finally activated using a sigmoid function

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.

3) 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 multispectral 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 $3 \cdot 10^{-}3$.

E. Software and Hardware

The code for all the experiments runs on a HP Pavillion laptop with a Intel I7 @2.8GHz processor. The available RAM is 16Gb, and the entire stack runs on CPU.

The software for the signal segmentation, processing, and the procedure to create the pseudoreal dataset is written in python, using a modified version of the repository EEGWave which is publicly availalble in the CodeOcean platform [**?**]. It uses the MNE library [**?**] for segmenting the data and some operations with numpy [**?**] for signal processing procedures. On the other hand, the generation of the plot of the signals is written in C++ using OpenCV, and all the neural networks architectures are run using Tensorflow's API for C++, based on Benny Friedman's article [**?**]. The full code is public and can be found at https://github.com/shipupi/BciSift/. For the sake of replicability [**?**], all the software and data for this experiment is fully online available.

III. Results

A. Validation on the Pseudoreal dataset

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 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 [**?**].

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.

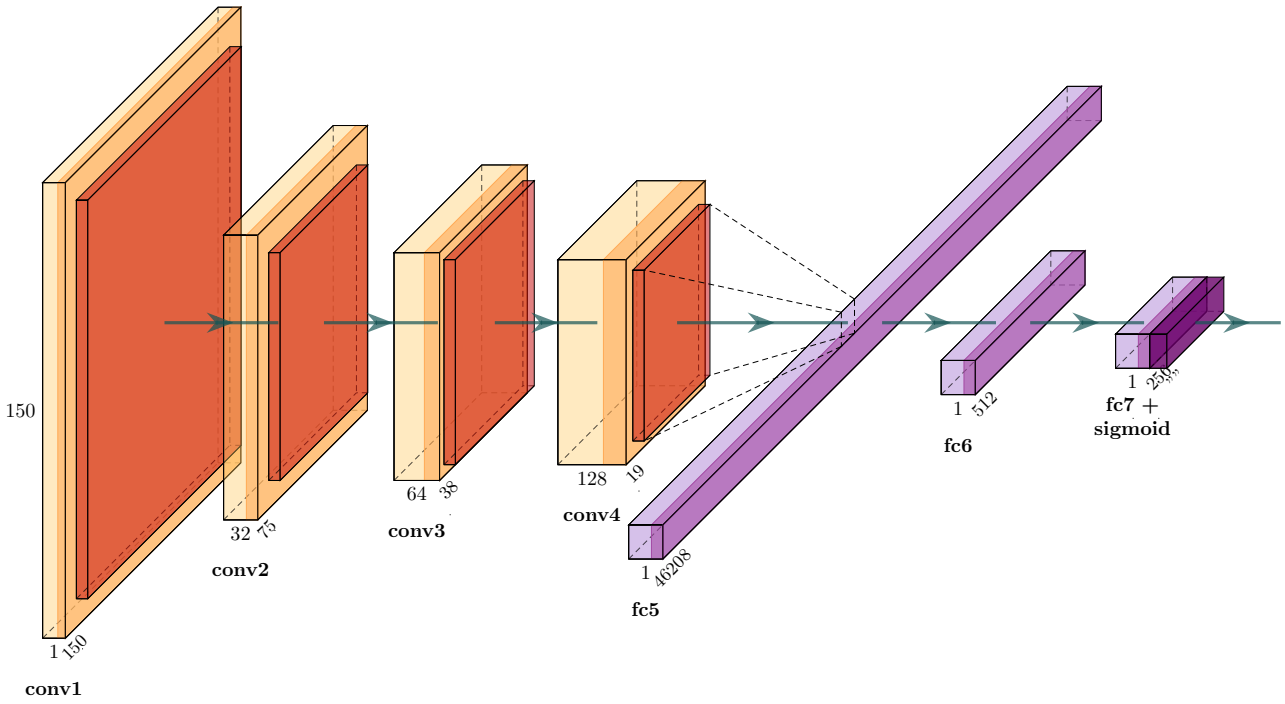


Fig. 3: SV16 is similar to VGG16, but has two less convolutional layers and two less dense layers

Subject	Original	HIST	xDAWN	VMD	EEGNET	VGG16	SV16	MSV16
1	98	35	99	99	99	15	10	0
2	98	85	99	99	99	70	50	75
3	98	25	99	99	99	30	30	40
4	98	55	99	99	99	30	40	30
5	98	40	99	99	99	35	50	50
6	98	60	99	99	99	45	40	50
7	98	80	99	99	99	70	65	80
8	98	95	99	99	99	90	95	100

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

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 [**?**]. 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,

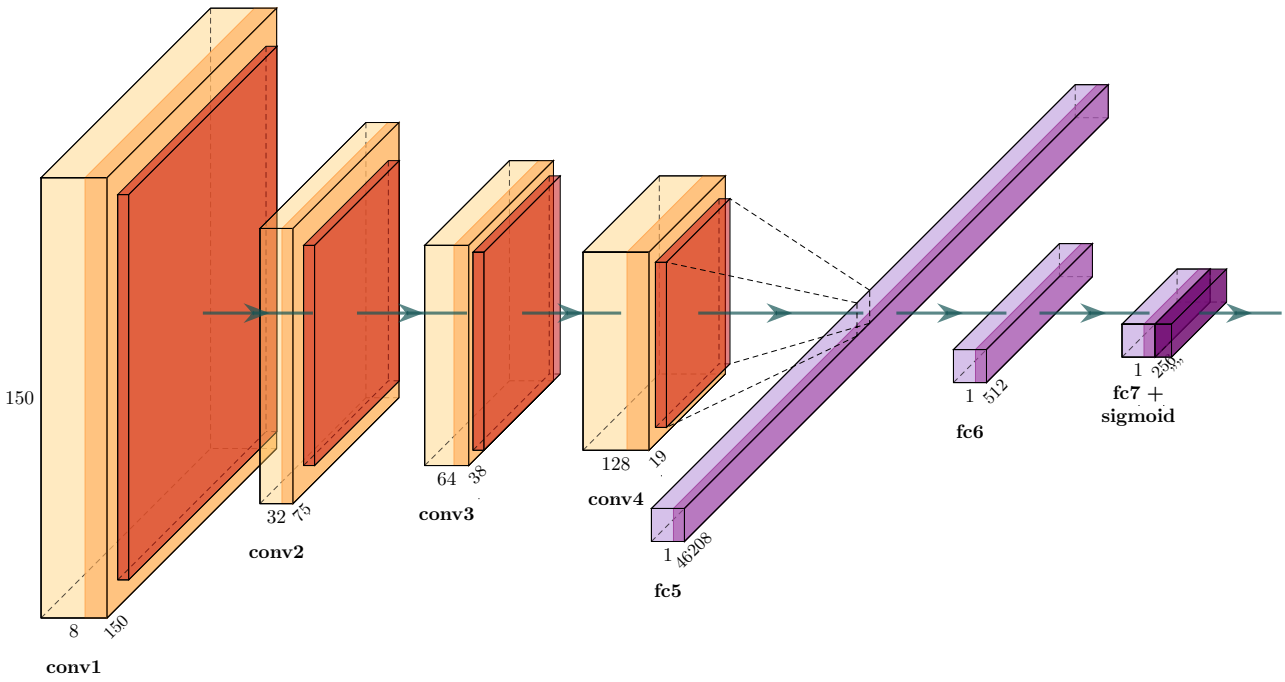


Fig. 4: MSV16 has the same architecture as the second one, but the input layer is modified for an 8-channel input

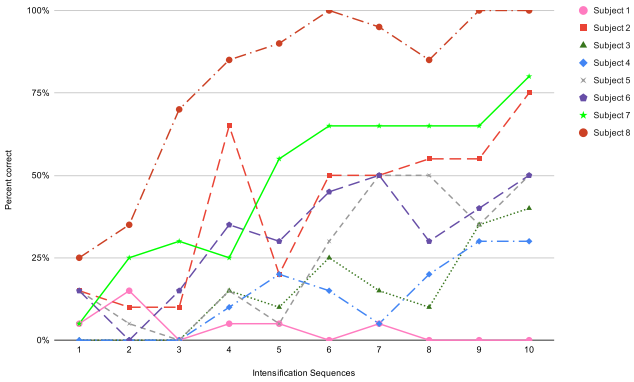


Fig. 5: Letter accuracy for MSV16 shows a significant improvement over SV16

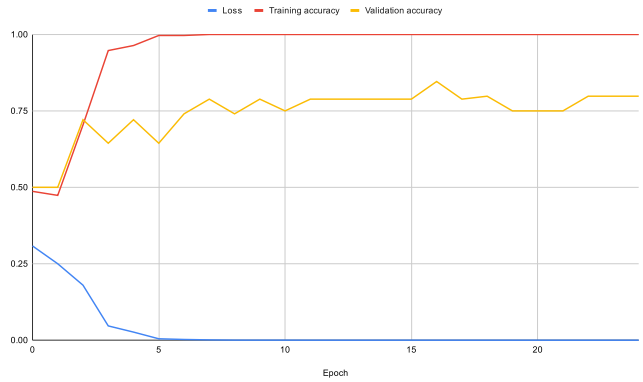


Fig. 6: Learning curve of subject 5, with an intensification level of 4 using multichannel training

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 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 **??**.

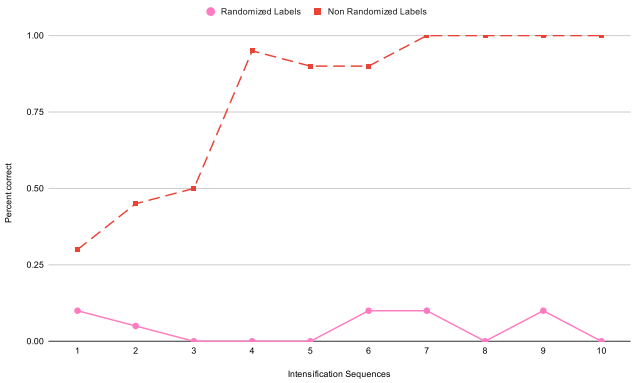


Fig. 7: Accuracy of MSV16 on subject 8, the red line shows a normal prediction using regular labels on the training set, while the pink line shows the predictions of the NN by training it with randomized labels on the training set. We can see that the accuracy on the pink line hovers around the *random*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

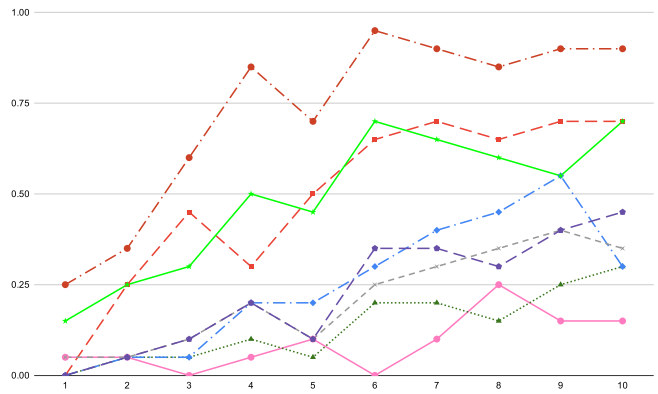
3) DL performance: Regarding the VGG16 architecture, we can see in figure **??** that the main issue is overfitting. The training sets are being learned perfectly, reaching a 100% accuracy, but the accuracy on the validation set does not increase, and oscillates around 60%, while this is a bit better 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.

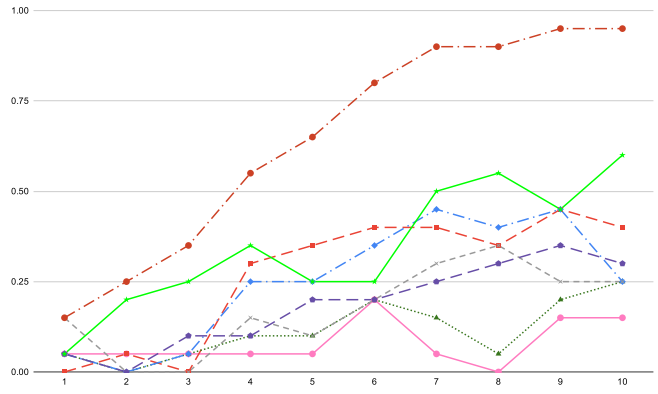
The biggest breakthrough of this work however is the MSV16 version, the inclusion of all the EEG channels into a single image, generated substantial improvements compared to both previous versions, and it even surpassed other proven methods such as SVM and HIST on some of the subjects, proving that using 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

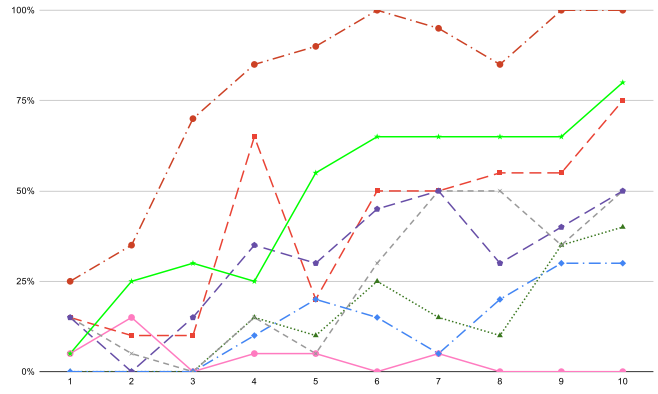
We would like to thank ITBA grants



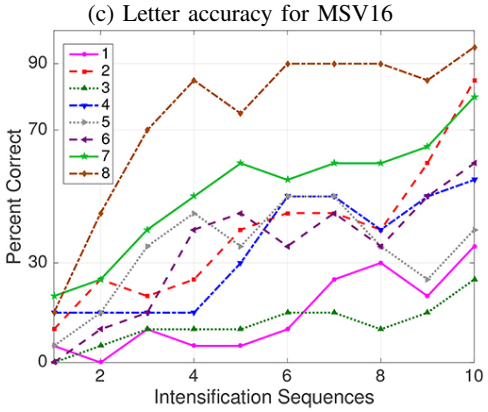
(a) Letter accuracy for VGG16



(b) Letter accuracy for SV16



(c) Letter accuracy for MSV16



(d) Letter accuracy for HIST method [**?**]

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 **??**