Contents

Table of Figures

Abstract

In the era of brain computer interfaces, the importance of P300 speller is evident. However, the Speller technology can have a wide foot print, only if its cost is reduced. Electroencephalogram (EEG) device is the major contributing factor to the cost of the speller. Our effort is directed towards using a low cost EEG device and to study if it can be used for the speller. We have taken the Farwell and Donchin's speller model as our base. This model uses a 6x6 key matrix. Previous studies around this model, use 8 or more sensors to correctly identify the target key. However since our low cost EEG device (Muse 2) has 4 sensors, we thought of beginning with a smaller (3x3) key matrix. Reducing the number of keys increases the probability of identifying the target key correctly. This study aims towards doing a comparative study of "accuracy" and "speed" of different sizes of BCI spellers using low cost, fewer sensors EEG device. This study is conducted on OpenViBE platform. It uses different classifiers (LDA, MLP and CNN) to process the EEG signals for target identification. This study also aims to identify the amount of training data required for speller and its consequent efficient classifier.

Introduction

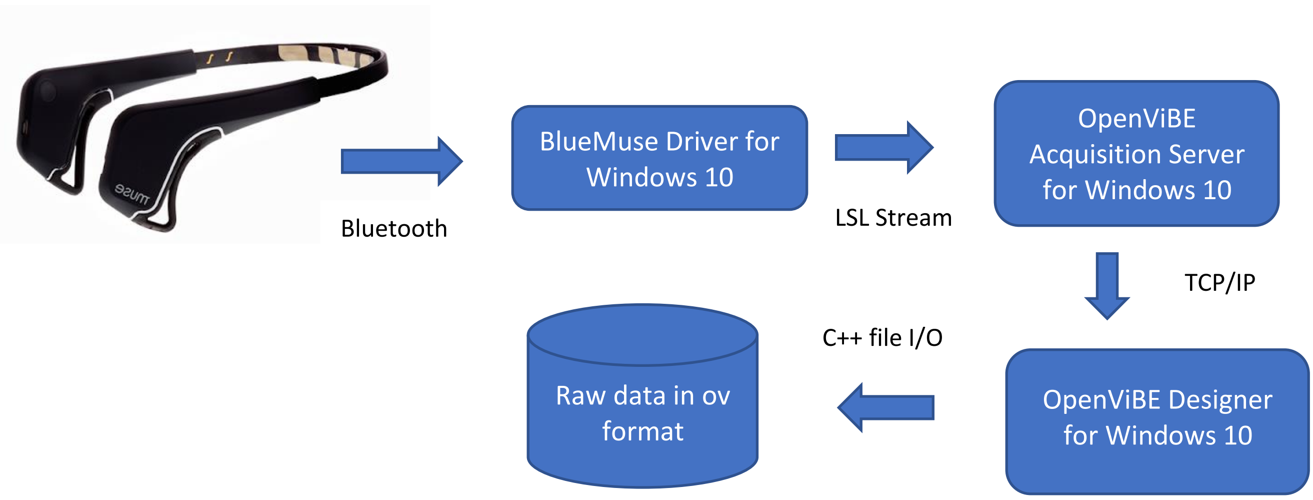
The Farwell and Donchin P300 speller is one of the important Brain Computer Interfaces used mainly for patients with Locked-In syndrome, who possess good cognitive ability. The speller is based on Oddball paradigm where the occipital lobe (the visual cortex) of the brain produces a P300 visual evoked potential, on encountering an infrequent deviant (target) stimulus among a sequence of similar repetitive stimuli. The name P300 refers to the fact that the peak of the signal appears after 300 msec on presenting the deviant stimulus to the subject. We are using Muse 2 device for our experiments. This device has two electrodes (AF7 and AF8) placed on the forehead and (TP9 and TP10) placed behind the ears as per 10-20 electrode positioning system. Muse 2, has no electrode placed on the occipital lobe. However, there are a couple of studies showing the presence of this visual evoked potential on Temporal and Frontal electrodes. Hence we decided to use this device for speller. We used OpenViBE platform for data generation, signal processing and prediction functionality. In order to take advantage of full range of TensorFlow's machine learning libraries, we extended the OpenViBE's python interfacing API and used Sklearn/TensorFlow's LDA, MLP and Conv1D classifiers and predictors. OpenViBE's speller user interface is used in order to generate the Oddball stimuli. The built-in speller user interface contains 6x6 matrix of characters and numbers. However, it is customized to contain 3x3 matrix of numbers. This paper will present a comparative analysis of 3x3 and 6x6 matrix performance with Muse device.

Experimental Setup

The experiment begins with interfacing Muse – 2 with OpenViBE using BlueMuse driver and collecting data from various subjects. The data collection process involves the following steps:

1. Muse -2 device communicates with the BlueMuse Driver over Bluetooth.
2. BlueMuse connects with OpenViBE's Acquisition Server over Lab Streaming Layer (LSL).
3. The Acquisition Server then sends the data to OpenViBE designer over TCP/IP.
4. OpenViBE designer captures the raw data in .ov format

Please look at the following figure to have a pictorial view of the above steps.



Below, we have shown the configuration of all the following components required for the project:

1. BlueMuse Driver
2. Acquisition Server
3. OpenViBE designer

Let's begin with BlueMuse Driver configuration.

Configuration of BlueMuse Driver

1. When the bluetooth connection with Muse-2 device and the BlueMuse driver is alive, the screen in Figure 1 appears on opening the BlueMuse driver user interface.
2. On clicking the **Gear Icon** in the top menu bar in Figure 1, the screen in Figure 2 appears, where we have kept all default setting intact, except we have unchecked all the checkboxes except the **EEG Enabled** checkbox.
3. Once the configuration in Figure 2 is done, we can start streaming EEG to OpenViBE acquisition server.

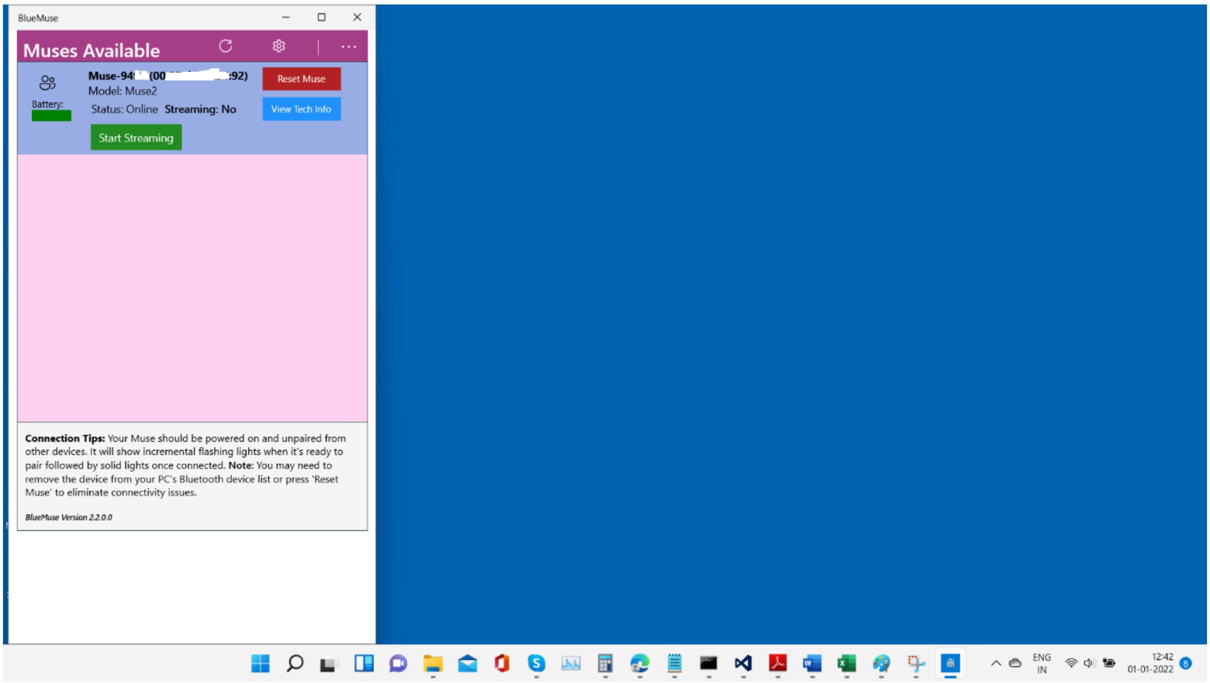


Figure 1 Streaming from Muse 2

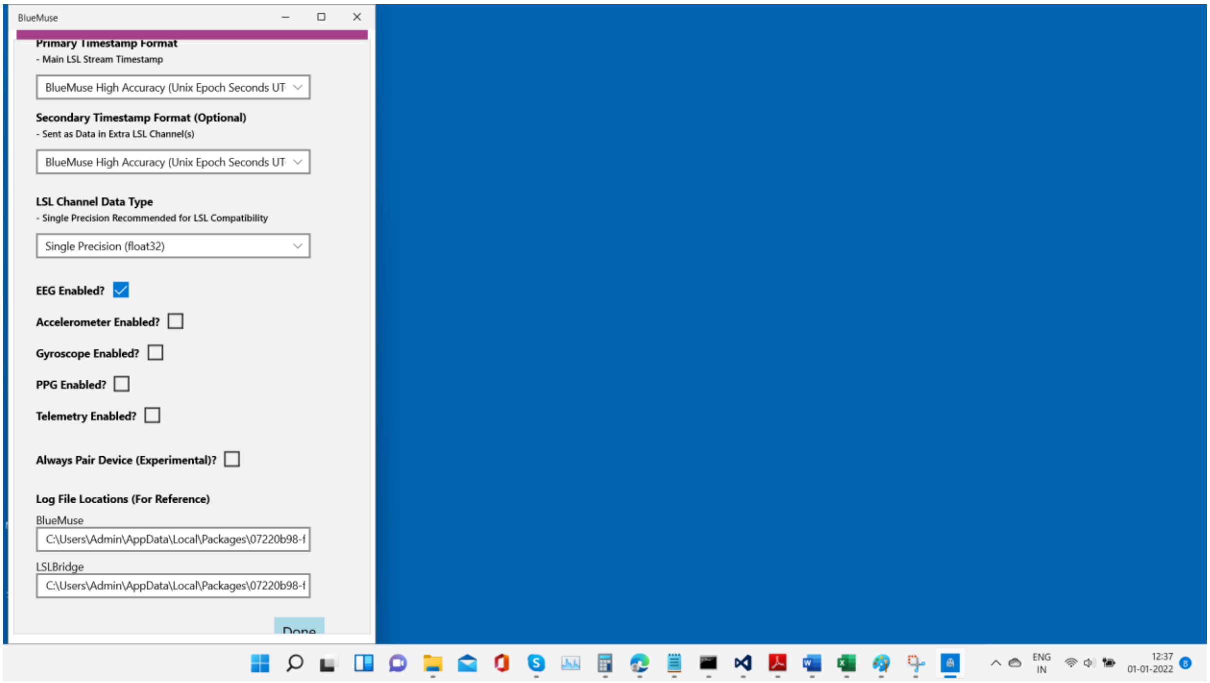


Figure 2 Muse 2 Driver Configuration for Streaming EEG

Configuration of OpenViBE Acquisition Server

1. Run the OpenViBE acquisition server to get its user interface as shown in Figure 3. Keep the default settings, except choosing **the LabStreamingLayer** as the communication medium between BlueMuse and Acquisition server.
2. Click on the **Driver Properties**. If things go smooth so far, **EEG** option will be shown in the signal stream box along with the device identity as shown in Figure 4. Click on Apply.
3. Click on **Preferences** button in Figure 3 and the user interface will appear as shown in Figure 5. Select the options/values as shown in the Figure 5. Click on Apply to go back to Figure 3.
4. In Figure 3 click on **Connect** and **Play** to get the OpenViBE acquisition server running with LSL driver.

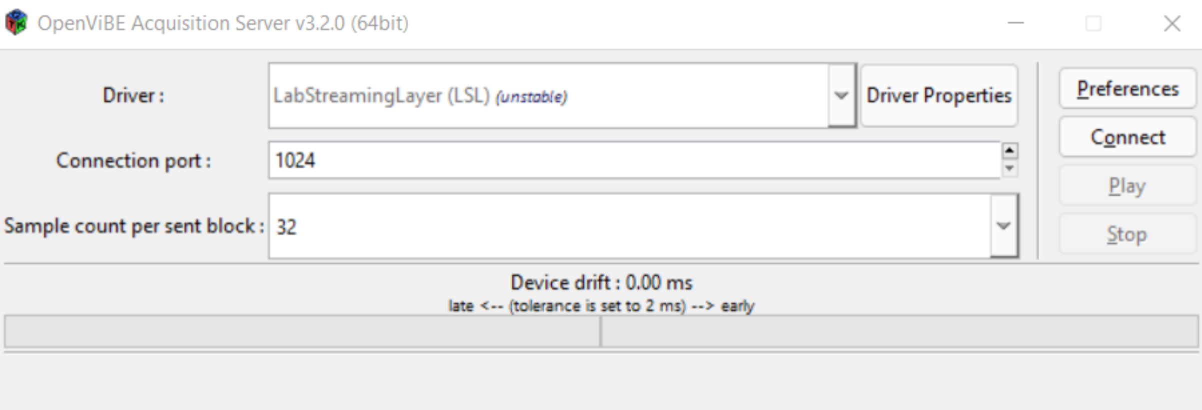


Figure 3 Configuration of OpenViBE acquisition server

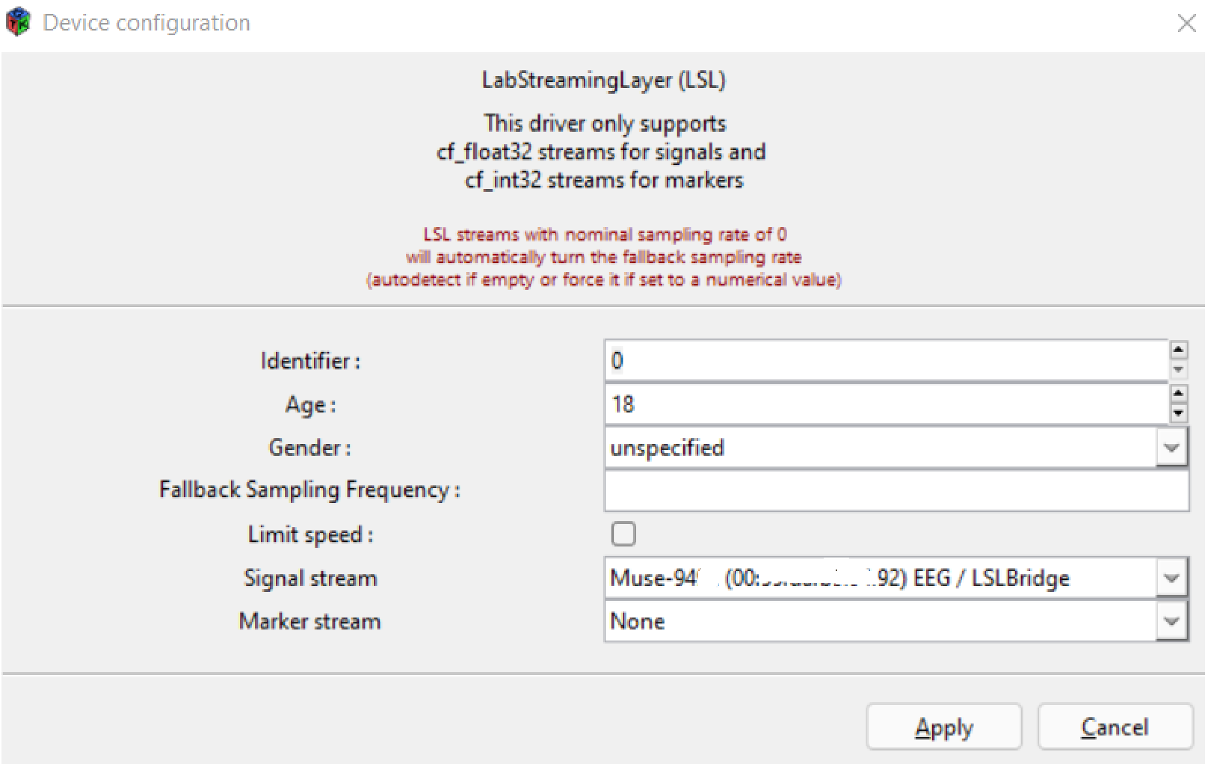


Figure 4 OpenViBE Acquisition Server Configuration - LSL

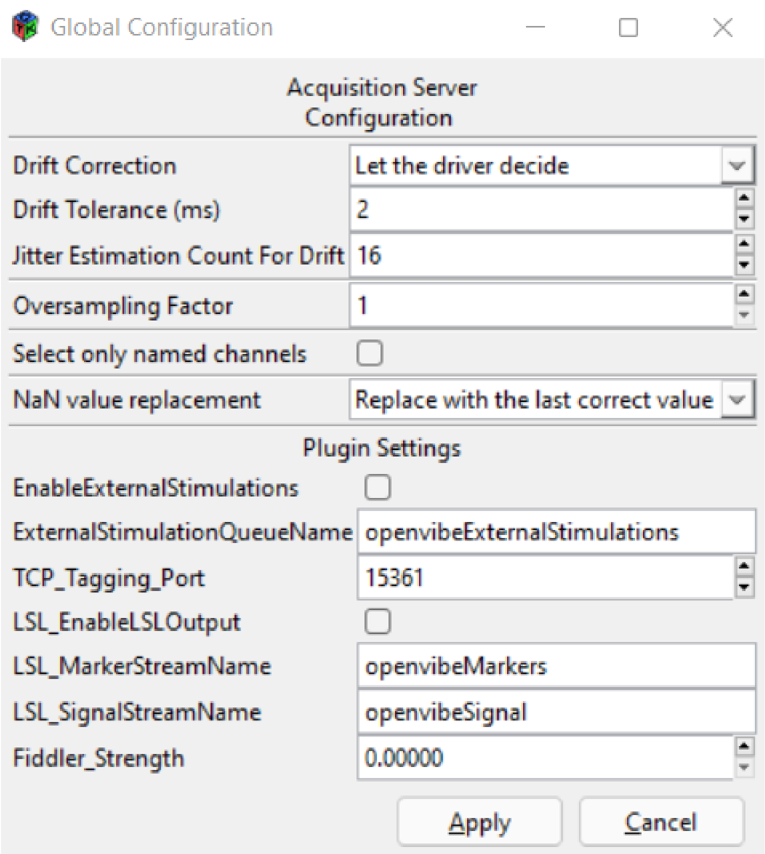


Figure 5 Acquisition Server Preferences

Configuration of OpenViBE Designer For Data Collection

Open the OpenViBE designer UI and open the **p300-speller-1-acquisition.xml**. Configure the **P300 Speller Stimulator** box as shown in the below figure (Figure 6). We have configured the flash duration to be 170 msec and non-flash duration to be 30 msec. This makes the stimuli repetition interval to be 200 msec. The number of rows in the speller UI are configured to be 3. Number of columns of the speller are also 3. This gives a 3 x 3 matrix of keys for the speller. We have configured each session to have 10 characters in the training session. When we run this scenario, a 3 x 3 speller looks as in Figure 7. Each of the training characters (target character) is highlighted in BLUE colour. Once the character gets highlighted, the subject has to focus on the character till it completes 24 flashes. The number 24, attributes to **12 repetitions x (1 row flash on target + 1 column flash on target = 2 flashes)**. After completing 24 flashes on the target, another character is presented to the subject in BLUE colour. Once all 10 characters are completed, the Speller User Interface closes and a data file is created in **.ov** format. A sample storage format is **p300-online-[2021.12.31-10.33.34].ov**. Rename this file such that files for different subjects can be identified correctly, for example: **Subject-1-p300-online-[2021.12.31-10.33.34].ov**. This file further needs to be processed to extract the epoch data for each of the stimulation.

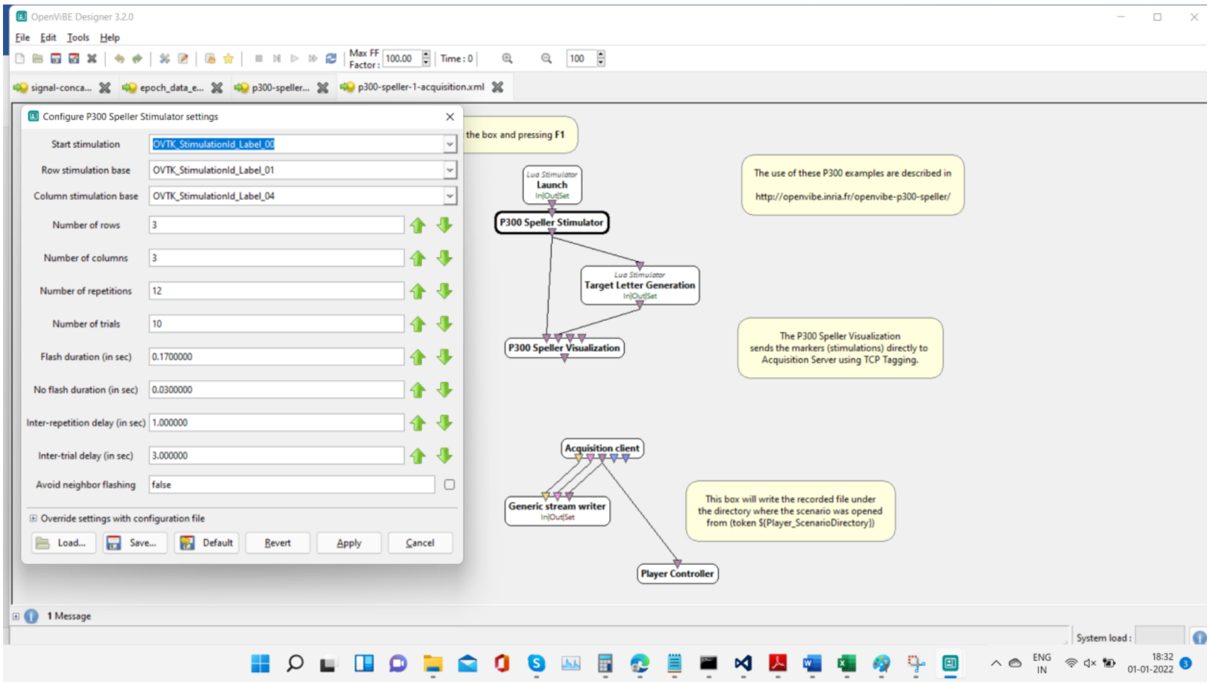


Figure 6 Data Acquisition Scenario of OpenViBE Designer

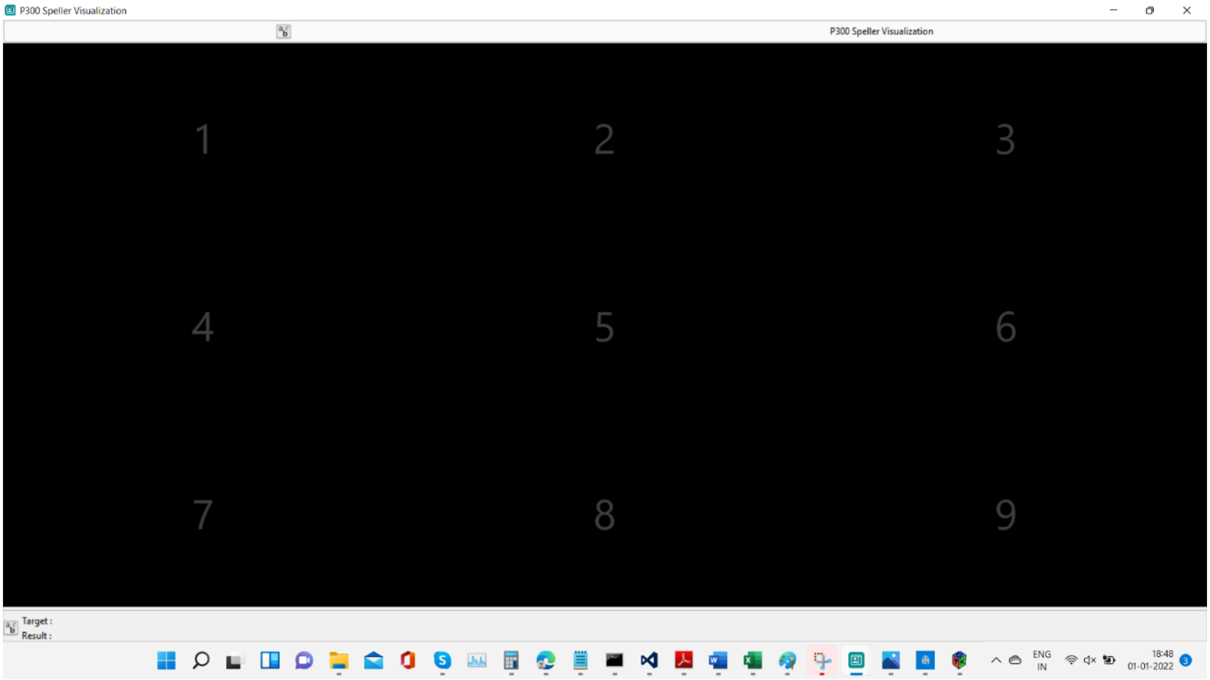


Figure 7 P300 Speller of size 3 x 3

Epoch Data Extraction

In the OpenViBE designer UI, open the scenario file **epoch_data_extraction.xml** (Figure 8). Configure the EEG channels (Figure 9) from which to extract the data, using the channel selector. In our condition Muse-2 provides four channels TP9, AF7, AF8 and TP10. Then configure the Temporal Filter as shown in Figure 10 for extracting the P300 signal. Most of the literature specifies a frequency band of 1 Hz to 12 Hz for extracting the P300 signal. Hence we have extracted the same band from the raw data using 4^{th} order Butterworth Bandpass Filter. The ripple is set to its default configuration by OpenViBE. The signal is then decimated by 8 using Signal Decimation Box (Figure 11). The epochs of 600 msec are extracted posterior to each stimulation (Figure 12). This duration of epoch is chosen based on the literature for extracting P300. A python script is written as per OpenViBE instructions, for dumping the epoch data into an excel sheet (in the **D** drive). This script is kept at the below location.

epoch extraction python script

Each row of the excel sheet identify samples of either target or nontarget epoch. Each row contains 76+1 columns. Out of which the first 19 columns of each row correspond to channel TP9, the second 19 columns correspond to channel AF7. The third 19 columns correspond to channel AF8 and the fourth 19 columns correspond to channel TP10. The last column of this excel sheet identifies whether the row belongs to target or non-target flash. Target flash is identified by a 1 and non-target flash is identified by a zero.

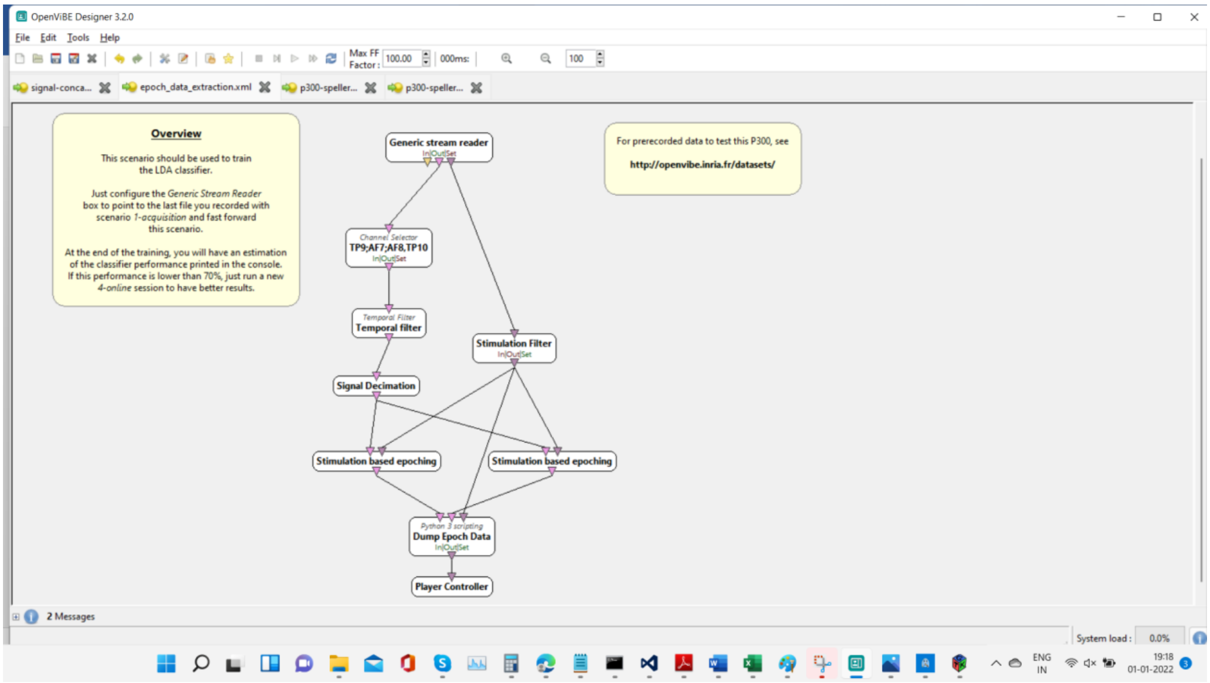


Figure 8 Epoch Data Extraction Scenario

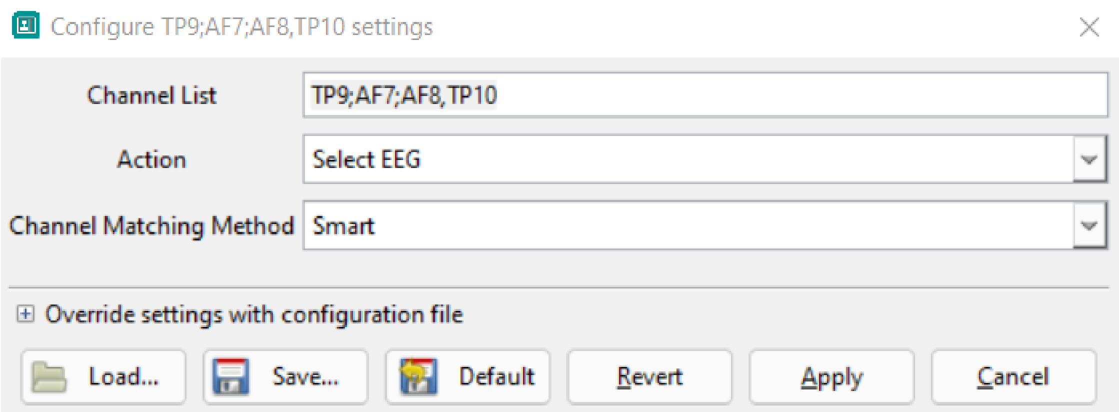


Figure 9 Configuration of Channel Selector Box

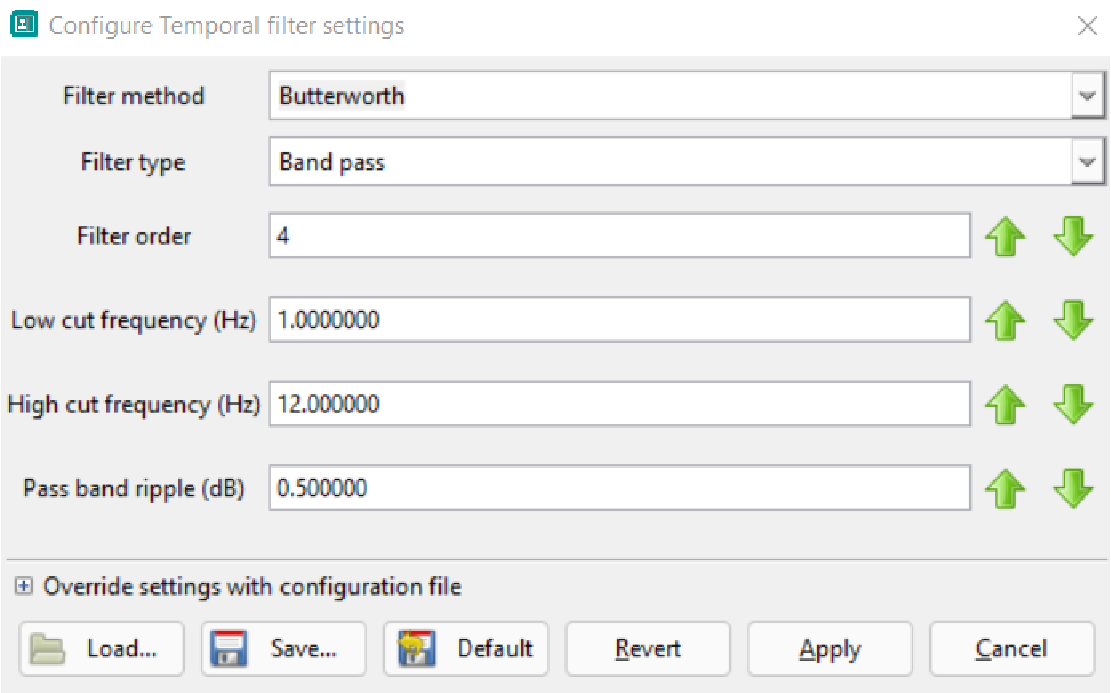


Figure 10 Bandpass Filter to Extract P300

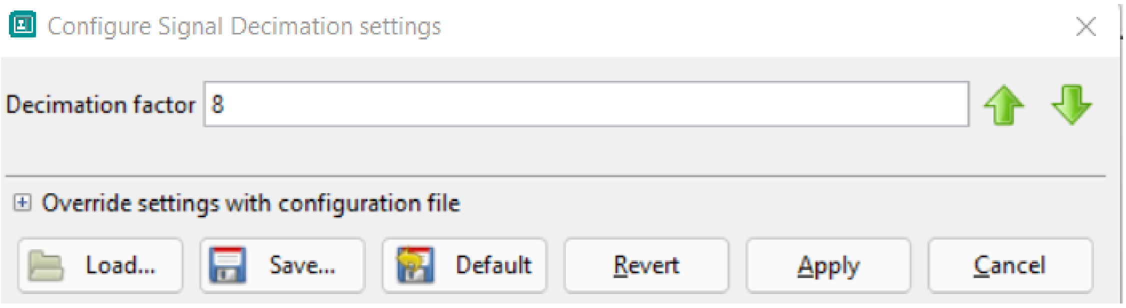


Figure 11 Decimation of Raw Data

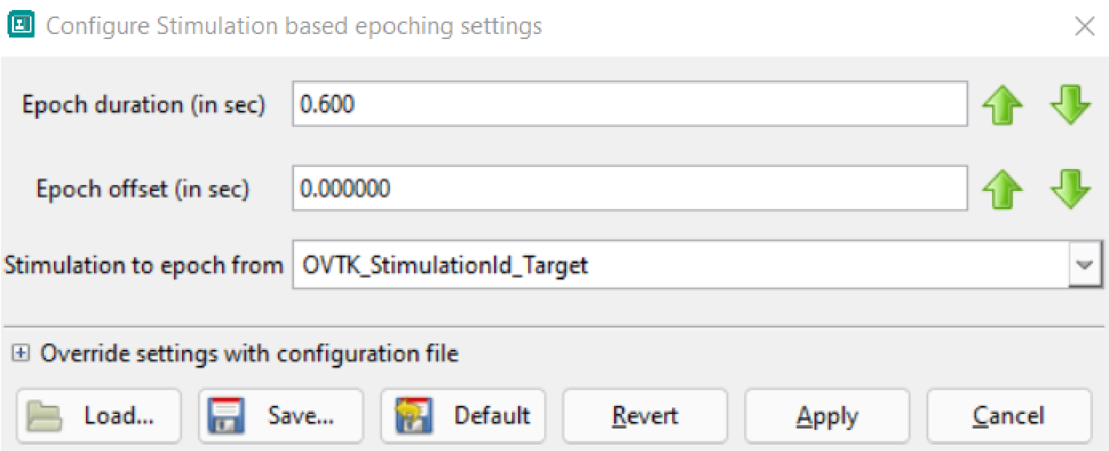


Figure 12 Set Epoch Duration

Training Data Collection

The models are trained with the data of 60 characters collected across two subjects. 40 characters belong to one subject (Subject 1 – 45 years) and 20 characters belong to the other (Subject 2 – 47 years). Each row and column flashes 12 times. This makes the target character flash 12 + 12 = 24 times. Rest non-target rows and columns (total 2 rows + 2 columns of non-target) also flash 12 times. This gives 12 x (2 + 2) = 48 non-target flashes for a target character. This gives total 24 x 60 = 1440 target flashes (epoch samples) in the excel sheet. It also gives 48 x 60 = 2880 non-target flashes (epoch samples) in the excel sheet.

Plot Average of Target Epochs and Check for P300

In this step, we calculate the feature by feature average (each feature correspond to a column in the **epochs excel sheet** generated in the above section) of the target and non-target epochs from the excel sheet. We plot this average and check whether target epoch average displays a P300 waveform. We also check whether the difference between target and non-target epochs is sufficiently clear. Below we have attached the python script for getting the epoch average.

[Script to Calculate Epoch Average]

This script calculates an epoch average and dumps it into another excel sheet with name **data_characteristics in the D drive**. The first row of the data_characteristics excel sheet corresponds to the average of Target epochs and the second row of the data_characteristics excel sheet corresponds to the average of Non-Target epochs. Figure 13 is the sample graph of two subjects with age 15 and age 47. For the subject of age of 15, the target epoch average shows a clear P300 potential on channel AF8. The peak occurs at location 45 (38 + 7). Which comes around at **220 msec** on channel AF8 after the start of stimulation. The peak amplitude is of 6 microvolts. For the subject of age 47, the target epoch average peak is rather unclear. The peak amplitude does not exceed beyond 2 microvolts in case of age group 45. We have shown the graph of P300 voltage variation across the age time span from the literature in Figure 14.

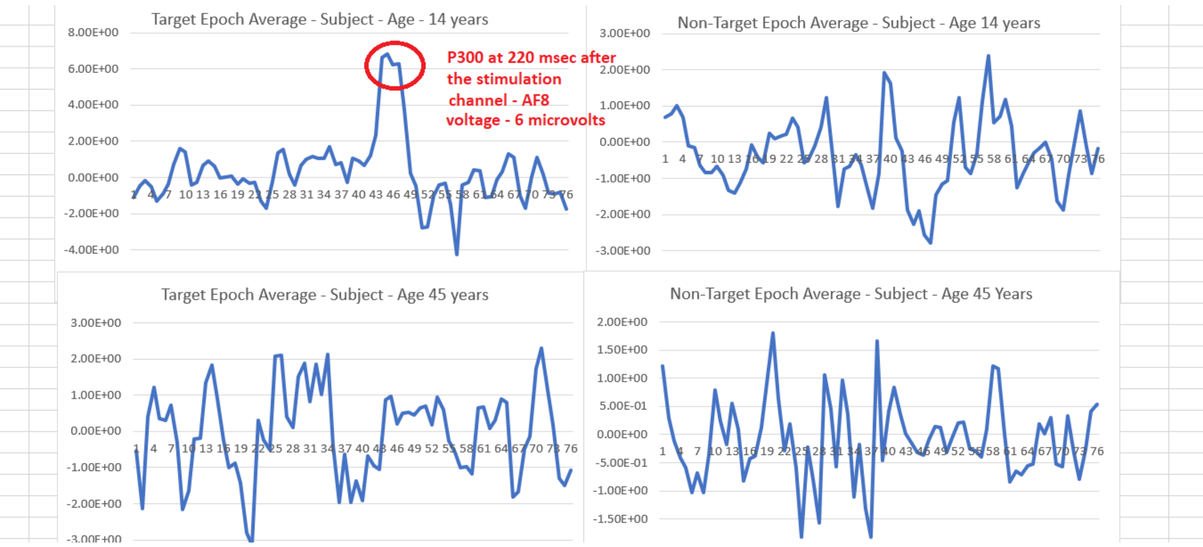


Figure 13 Target and Non-Target Epoch Average

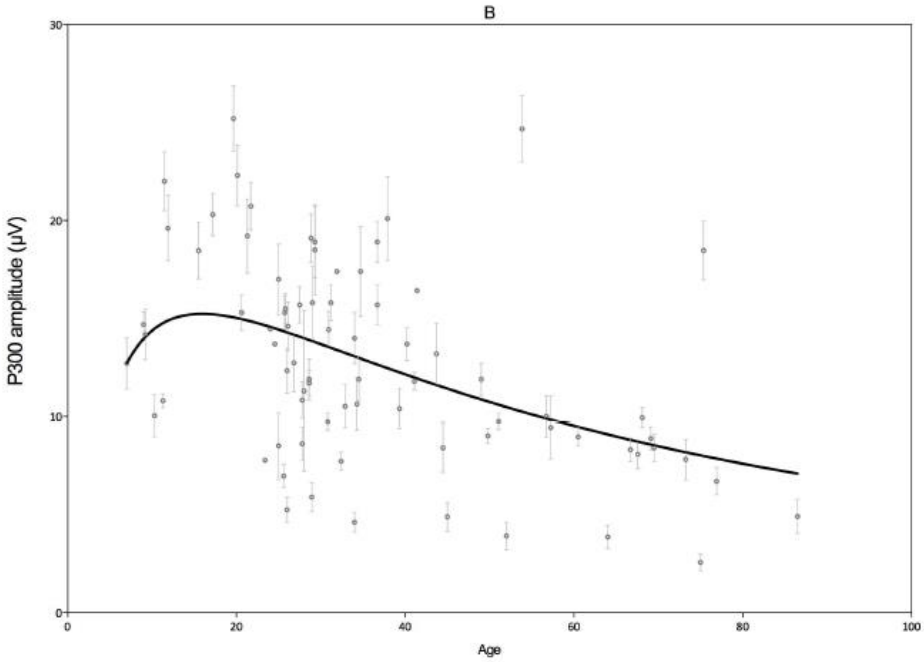


Figure 3. P300 latency and amplitude trajectories across the lifespan as obtained from the meta-analysis. Dots represent (subgroups from a) study. Error bars represent SEM.
doi:10.1371/journal.pone.0087347.g003

Figure 14 P300 voltage variations across age span

Online Data Collection

There are two subjects for whom we have gathered the online data (Subject 2 – 47 years) and (Subject 3 – 15 years). Online data collection requires a model to predict the characters. This model is created using the training data and used for prediction. We used 10 characters for subject 2, and 10 characters for subject 3 for online sessions. It gives us 24 x 10 = 240 target online flashes and 48 x 10 = 480 non-target flashes. We expect the following confusion matrix for online sessions after ideal prediction:

$$\begin{bmatrix} 480 & 0 \\ 0 & 240 \end{bmatrix}$$

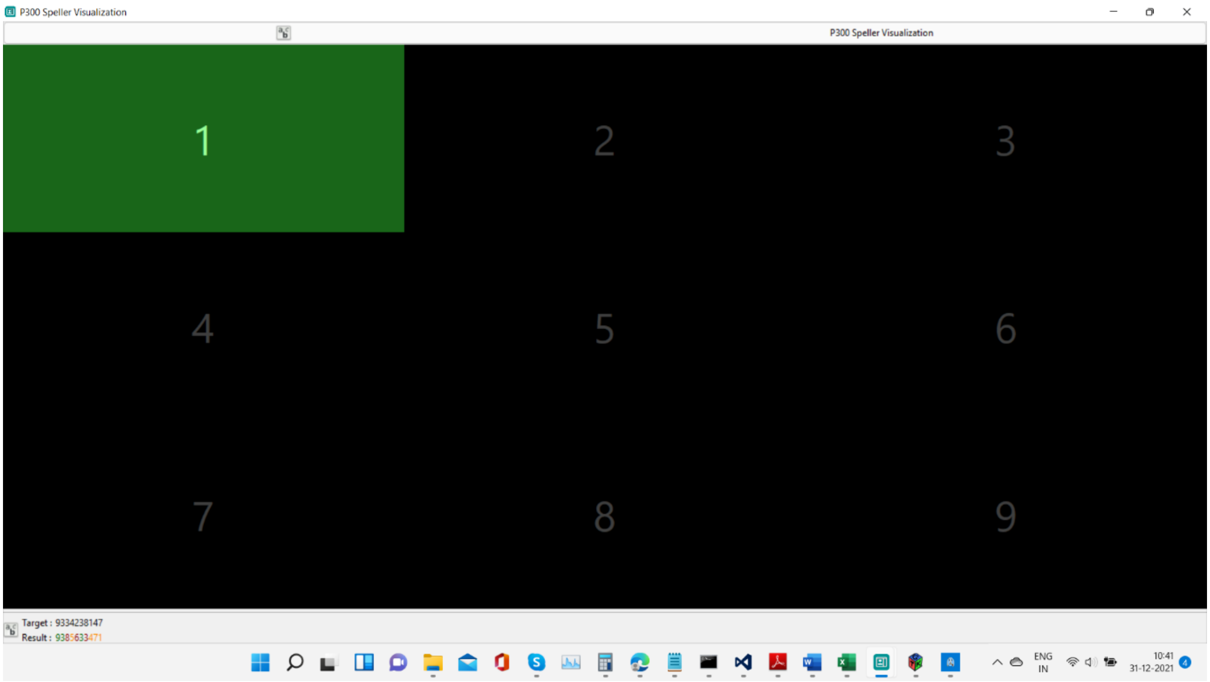


Figure 15 OpenViBE Prediction Speller UI

Create a TensorFlow Model with LDA, MLP and CNN

We created TensorFlow models with the extracted target and non-target training data in the section above. Below are the python/TensorFlow scripts for LDA, MLP and CNN classifier model creation. The verification scripts are used for predicting the online data and rendering the confusion matrix.

Conv1D model creation script

MLP model creation script

LDA model creation script

Conv1D model verification script

MLP model verification script

LDA model verification script

Pre-processing steps in model creation

1. Non-Target epochs are duplicated for data augmentation
2. The Target and Non-Target epochs are then balanced, to have the probability of existence of target and non-target both to be 0.5. (The original data in the epoch extraction step has target and non-target proportion is 1:2)
3. 70% data is used for training and 30% data is used for validation
4. The mean of all the training samples is subtracted from all the training samples and thus the data is transformed to be **zero mean**
5. All the samples which are above 60 microvolts are then saturated to a value of 60
6. All the samples which are below -60 microvolts are then saturated to a value of -60
7. The data is then normalized to be in the range from [0 1]. The formula used for normalization is below.

$$normalizeddatapoint = \frac{\left(datapoint + abs\left(min(data)\right)\right)}{\left(max(data) - min(data)\right)}$$

8. The Sklearn/TensorFlow LDA, MLP and CNN models are created. For LDA, we have used SVD as a solver. For MLP we used Adam optimizer with max iterations of 1200. Alpha regularization set to 0.001 and two hidden layers were used first one with 100 neurons and the second one with 30 neurons. The CNN model is created as below, because it is easier to understand it with code. We came up with this model after reading literature available over CNN models for EEG and doing some trials.

```
model=Sequential()

model.add(Conv1D(filters=20, kernel_size=4, padding='valid', activation='relu', strides=1,
input_shape=(19, 4)))

model.add(Conv1D(filters=20, kernel_size=4, padding='valid', activation='relu', strides=1))

model.add(AveragePooling1D(pool_size=2))

model.add(Dropout(0.2))

model.add(Conv1D(filters=40, kernel_size=3, padding='valid', activation='relu', strides=1))

model.add(GlobalMaxPooling1D())

model.add(Dense(20, activation='relu'))

model.add(Dense(2, activation='softmax'))
```

9. The models' training time is around 60 seconds

10. After training the model, we have used each model for prediction and we are getting the following results.

Results - Classifier Performance

We know that our models are trained with the data, where both the subjects are of age above 45. The P300 peak is not clearly visible for both. When this model is used for 3 x 3 key prediction, the subject with age 14 got 50% online accuracy and the subject with age 45 got 45% accuracy. **It is clear from the confusion matrix that False Positives are twice greater than the True Positive in all the online scenarios.** However the prediction of actual keys in the 3 x 3 matrix is observed to be better, when the first three principal components show high variance. Also, the P300 peak is clearly visible when the PCA component variance is high. Please check the online key prediction accuracy for the two subjects in the table given below.

Table 1 Online Key Prediction Accuracy

	Key predictions	Prediction Accuracy
Subject – Age 14 years	3 green characters + 4 orange characters + 3 red characters	3 x 10 + 4 x 5 + 3 x 0 = 50 percent
Subject – Age 47 years	1 green characters + 7 orange characters + 2 red characters	1 x 10 + 7 x 5 + 2 x 0 = 45 percent

The below table shows the performance of the three classifiers that we have used, for prediction of Online data.

Table 2 Classifier Performance and PCA component variance

Classifier	LDA	MLP	Conv1D (CNN)	PCA – 1 target component variance	PCA – 2 Target component variance	PCA – 3 Target component variance
Subject – Age 14 years	[[212 268] [120 120]] Online acc: 46%	[[295 185] [143 97]] Online acc: 54%	[[254 226] [139 101]] Online acc: 49%	24%	20%	7%
Subject – Age 47 years	[[246 234] [117 123]] Online acc: 51%	[[234 246] [121 119]] Online acc: 49%	[[264 216] [130 110]] Online acc: 51%	15%	9%	8%

Conclusion

Our goal should be to get subjects, for whom the first few principal components show a high variance. This will result the prominent appearance of P300 potential. With this data, we should be able to generate a good model. This model should be able to correctly classify the epochs and hence should predict correctly. The best online prediction accuracy we have achieved so far is 50%. We should increase the accuracy of 3 x 3 key matrix up to 70%.

Limitations

1. Configuration of a 3 x 3 speller and 6 x 6 speller in OpenViBE is out of scope of this document
2. The results of 6 x 6 matrix need to be put in the format of Table 1 and Table 2
3. The OpenViBE prediction scenario is yet to be covered in detail

References

- Figure 1 derived from OpenViBE (inria.fr)

- Figure 14 derived from https://doi.org/10.1371/journal.pone.0087347

- Configuration of OpenViBE 3 x 3 key matrix Starting from ZERO on Brain-Computer Interface (BCI) — Openvibe BCI-Speller XDawn | by Apiporn Simapornchai | Medium

- OpenViBE P300 Speller tutorial / questions — OpenBCI Forum

- Presence of P300 potential on forehead - A Brain-Computer Interface–based P300 Speller for Home Appliances Control System by Praveen Shukla, R. K. Chaurasiya, Shrish Verma

- Presence of P300 potential on forehead - https://alexandre.barachant.org/blog/2017/02/05/P300-with-muse.html

- Interstimulus interval and epoch duration - Overlap and refractory effects in a brain–computer interface speller based on the visual P300 event-related potential by SMMMartens, N J Hill, J Farquhar and B Sch¨olkopf

- Preprocessing of EEG – An efficient P300-based brain-computer interface for disabled subjects Ulrich Hoffmann; Jean-Marc Vesin, Touradj Ebrahimi, Karin Diserens