UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

BACHELOR THESIS

Biomedical Engineering Degree

BAD CHANNEL DETECTION IN MEG RECORDINGS

Author: Alejandro Villalba de la Arada
Director: Jürgen Dammers
Co-Director: Beatriz Giraldo

May 2021

Resum

En la actualitat el camp del preprocessament de senyals és un àmbit que es troba en constant desenvolupament. Això és degut a la creixent utilització dels senyals biològics en diferents àmbits de la recerca i de la medicina. El preprocessament d'aquestes és indispensable per a poder extreure informació fiable i de qualitat. Una part d'aquest preprocessament consisteix en la detecció dels anomenats 'Bad channels'. Aquests són canals que contenen dades que no poden ser utilitzades per al posterior processat ja que contenen informació corrompuda. En aquest projecte es plantegen i es desenvolupen dos mètodes de detecció de canals dolents, a més d'una pipeline utilitzada per a analitzar l'efectivitat d'aquests i de qualsevol futur mètode.

Resumen

Hoy en día el campo del preprocesado de las señales es un ámbito en constante desarrollo. Esto es debido a la creciente utilización de las señales biológicas en diferentes ámbitos de la investigación y de la medicina. El preprocesado de estas es indispensable para poder extraer información fiable y de calidad. Una parte de este preprocesado consiste en la detección de los llamados 'Bad channels'. Estos son canales que contienen datos que no pueden ser utilizados para el posterior procesado ya que contienen información corrompida. En este proyecto se plantean y se desarrollan dos métodos de detección de canales malos, además de una pipeline utilizada para analizar la efectividad de estos y de cualquier futuro método.

Abstract

Nowadays the field of signal preprocessing is a constantly developing area. This is due to the increasing use of biological signals in various fields of research and medicine. The preprocessing of these signals is indispensable to extract reliable and high quality information. Part of this preprocessing involves the detection of so-called 'bad channels'. These are channels that contain data that cannot be used for further processing because they contain corrupted information. In this project, two bad channel detection methods are proposed and developed, as well as a pipeline used to analyze the effectiveness of these and any future methods.

Acknowledgments

This project would not have been possible without a group of people to whom I would like to give my most sincere acknowledgments.

First of all, I would like to thank my main guide Nikolas Kampel during the whole development of the project for his attention and patience, being day after day offering his knowledge at my disposal and guiding me on the right path. He has been able to bring out the best version of me when executing the ideas raised during the project. His dedication and motivation has been very inspiring.

Secondly, I would like to thank Jürgen Dammers for giving me the opportunity to join such a high level research group, as well as for taking care of my comfort and treating me as a member of the team.

Last but not least, I would like to thank all my family for making it possible for me to live abroad (in Germany) in order to develop this project so important to me and for supporting me unconditionally during the long development of this project.

Index

1. Introduction

One way to study and understand the functioning of the brain is through functional neuroimaging. This consists of recording certain brain signals and relating them to certain brain functions. There are several methods of functional neuroimaging, among them magnetoencephalography. This consists of recording the magnetic fields that arise in the brain from the electrical interactions between neurons. The recording is done by means of instruments called magnetometers. Because these signals are very weak, the magnetometers have a very high sensitivity that allows them to be recorded, but this fact also brings with it a disadvantage, and that is that, being a biological signal, it has a high degree of distortion and noise. This makes the subsequent analysis of these signals is difficult, since filtering and / or distinguishing noise from the signal to be analyzed is a process of high complexity or impossible in some occasions.

In these cases where it is not possible to filter or eliminate the noise is when we talk about bad channels. MEG signals are recorded, as mentioned above, by means of magnetometers. These recordings consist of multiple channels. When a channel cannot be interpreted or is not recorded correctly, it is marked as a bad channel.

The detection of bad channels is a very broad and constantly advancing field. This project is focused on the approach and development of methods to detect bad channels.

1.1. Context

This project has been developed at the research center Forschungszentrum Jülich (FZJ), an institution located in Jülich, Germany. They are in charge of leading multiple research projects related to various scientific fields, from health sciences to electronic sciences. It consists of 10 different institutes and this work was completed specifically at the Institute of Neuroscience and Medicine (INM-4), during a curricular internship.

1.2. Motivation

The need for quality signals to be able to perform studies correctly is indispensable. The main motivation of the project is to contribute not only to improve biomedical signal preprocessing, but also to improve its optimization and, therefore, the speed with which it is carried out.

1.3. Objectives

This project has two main objectives:

- The first objective is the design of two methods to find outliers. For this purpose, classification machine learning algorithms will be used, which, based on previously defined thresholds, will be able to distinguish bad channels among all the channels of the provided file.
- The second objective is to develop a pipeline capable of showing the efficacy of not only the methods proposed in the first method, but of any future method created for the same purpose.

2. State of art

In order to understand the methods developed in this project it is necessary to explain several concepts and points, as well as to contextualize the project by showing other ways of detecting bad channels that exist at the moment.

2.1. MEG

The signals analyzed in this project are MEG (magnetoencephalography) signals. To understand and explain what they are and what they consist of, the book 'MEG: An Introduction to Methods' has been consulted. (1)

2.1.1. Definition

Magnetoencephalography consists of recording the magnetic fields that occur in the brain from different activities or states. This, therefore, is considered a direct measurement of the activity of neurons.

The measurement system is non-invasive, which may present certain limitations when measuring the signal. Since these signals have to be recorded from the scalp, it is completely impossible to locate their exact origin when processing them. For this purpose, different highly complex models are used, which means that the results always have a certain degree of uncertainty, since they depend directly on the design of the model used.

2.1.2. MEG Recording

In order to make MEG recordings, very powerful and expensive equipment is required. The sensitivity of this equipment has to be very high, due to the weakness of the magnetic field. In addition, it requires a high signal isolation system (which is achieved not only by the hardware but also by the software) because biological signals are signals with a lot of interferences.

For the recording of these signals, field detectors called SQUIDs are necessary. These sensors are extremely sensitive field detectors. Their acronym stands for Superconducting Quantum Interference Device. They have an outer diameter of typically less than 1 mm and are currently used in sensor arrays covering the entire head with 100 up to 300 channels.(2)

The system used to record the files used in the project is the MAGNES 3600 system (4D Neuroimaging, San Diego, CA).

Figure 1. Magnes 3600 system (3)

This system has 23 reference channels, of which 18 are magnetometers and 5 are first order gradiometers, and 248 magnetometer channels.

Normally the signal recordings are not simply MEG, they are often combined with EEG, EKG and EOG recordings. This is because, as the sensitivity of the sensors is very high, any kind of biological signal alters the recording. Therefore, from heart palpitations to eye movements and eye blinks are reflected in the MEG and it is very useful to have these recordings for further processing.

2.1.3. Bad channels

Sometimes there are channels that contain corrupted information and therefore make it difficult to analyze the signals and study them. So, when a recording cannot be used for further processing, a channel is marked as bad.

There are three different ways of categorizing the methods for marking bad channels. In order to understand it correctly, first of all the concept of epoch has to be defined. Epochs are equivalent length chunks of a continuous signal. They are usually done around a particular event, such as around the time of a heartbeat.

The three groups are:

- Spatial bad marking: This consists of comparing the characteristics of an entire channel with the characteristics of all other channels.

- Temporal bad marking: It consists of dividing a channel into epochs and comparing the characteristics of each epoch within it, taking into account the amount of time that a channel can be bad.
- Bad epoch rejection: This third one is a combination of the two previous ones, comparing temporal characteristics between epochs and spatial characteristics between all channels.

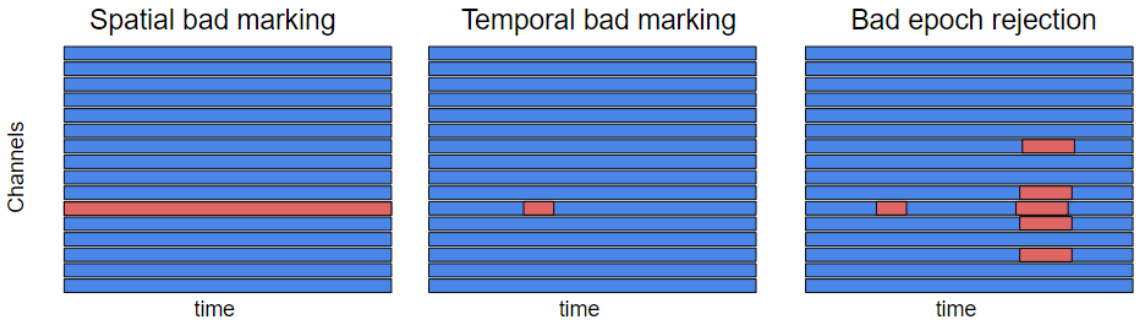

Figure 2. Types of bad channels categorization

Bad channels are very different across recordings and especially across different recording sites. Examples of bad channel types can be seen below, some being from HCP (Human Connectome Project) data and others from data recorded at the FZJ:

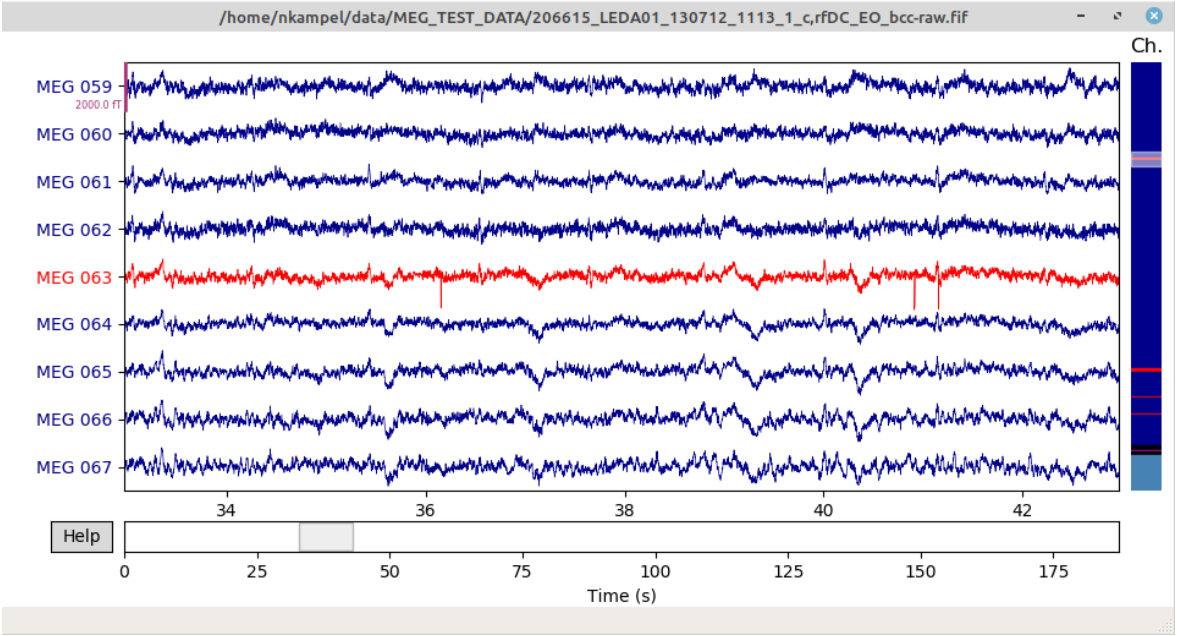

Figure 3. Low frequency jumps (FZJ data)

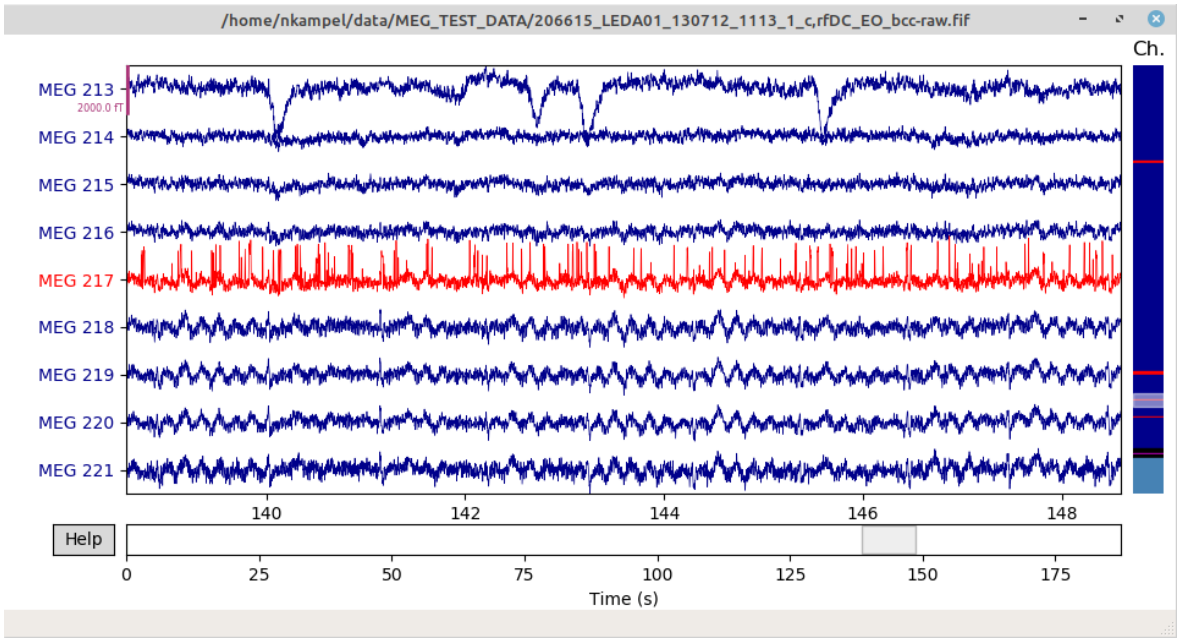

Figure 4. High frequency jumps (FZJ data)

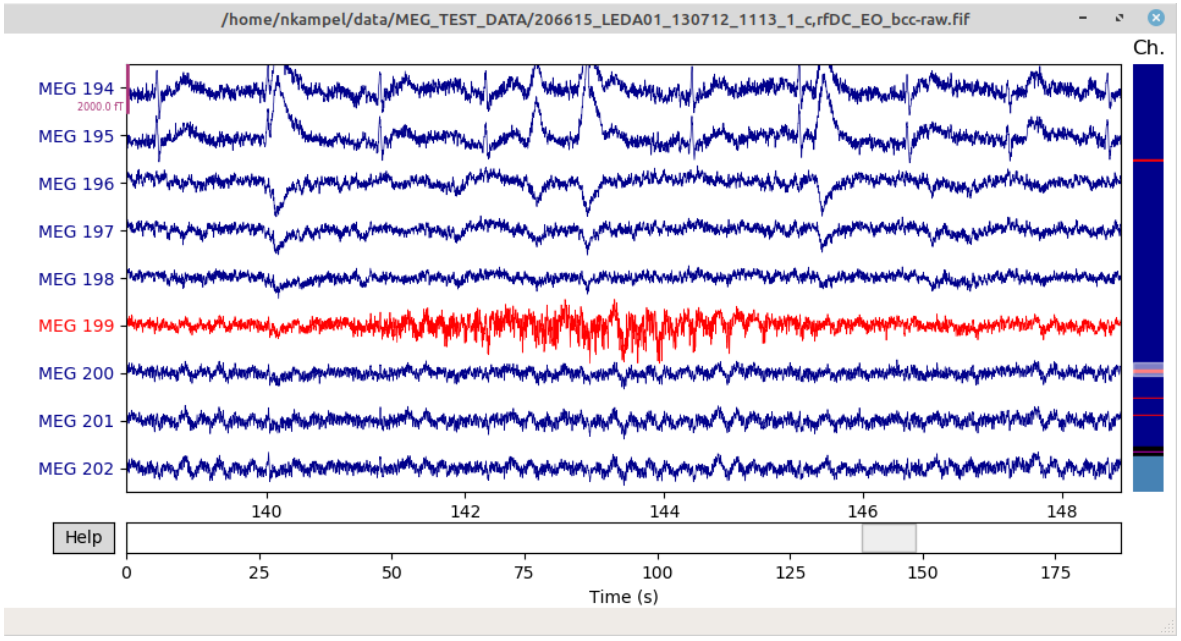

Figure 5. Noisy episode (FZJ data)

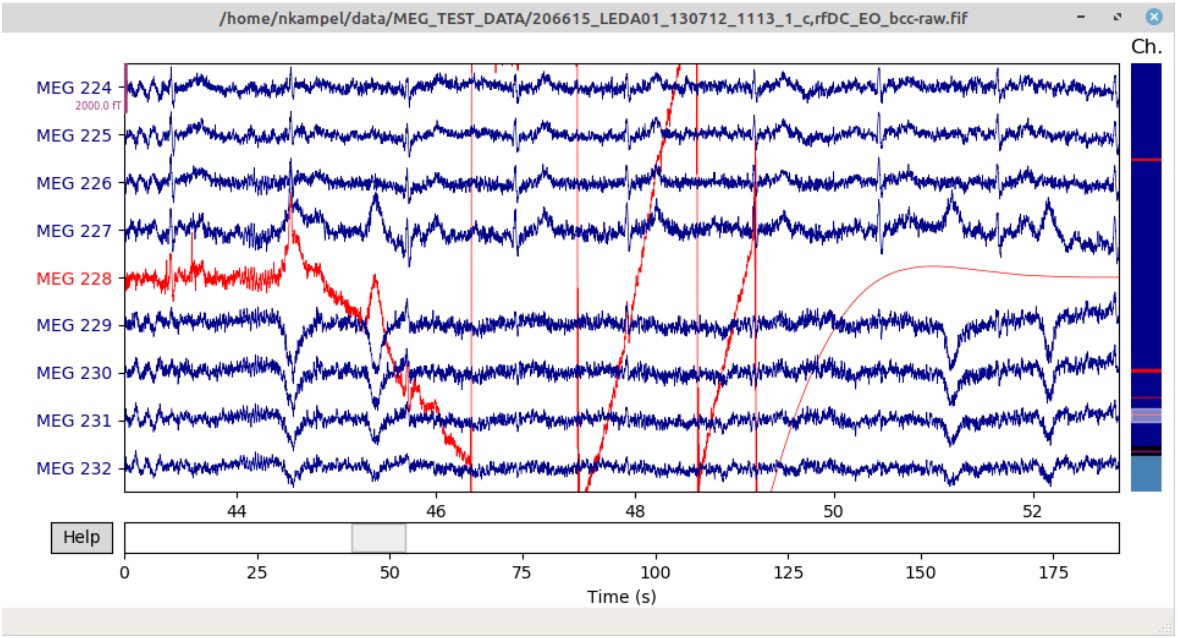

Figure 6. Sudden event + flat channel (FZJ data)

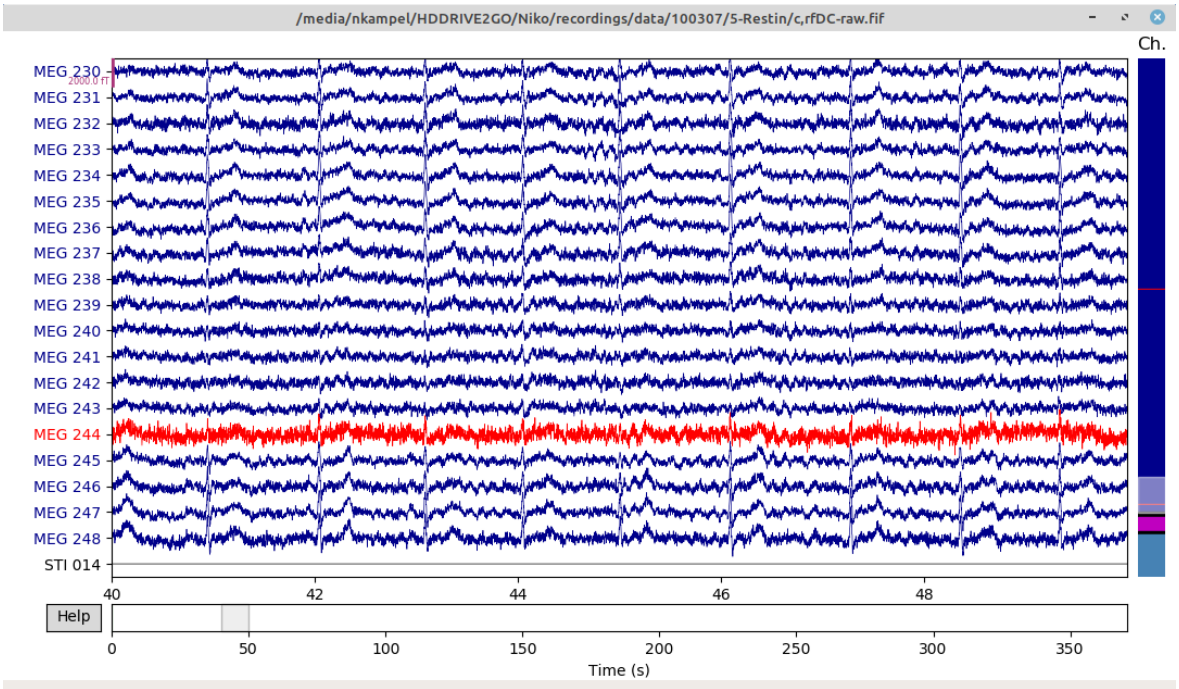

Figure 7. Slightly increased noise (HCP Data)

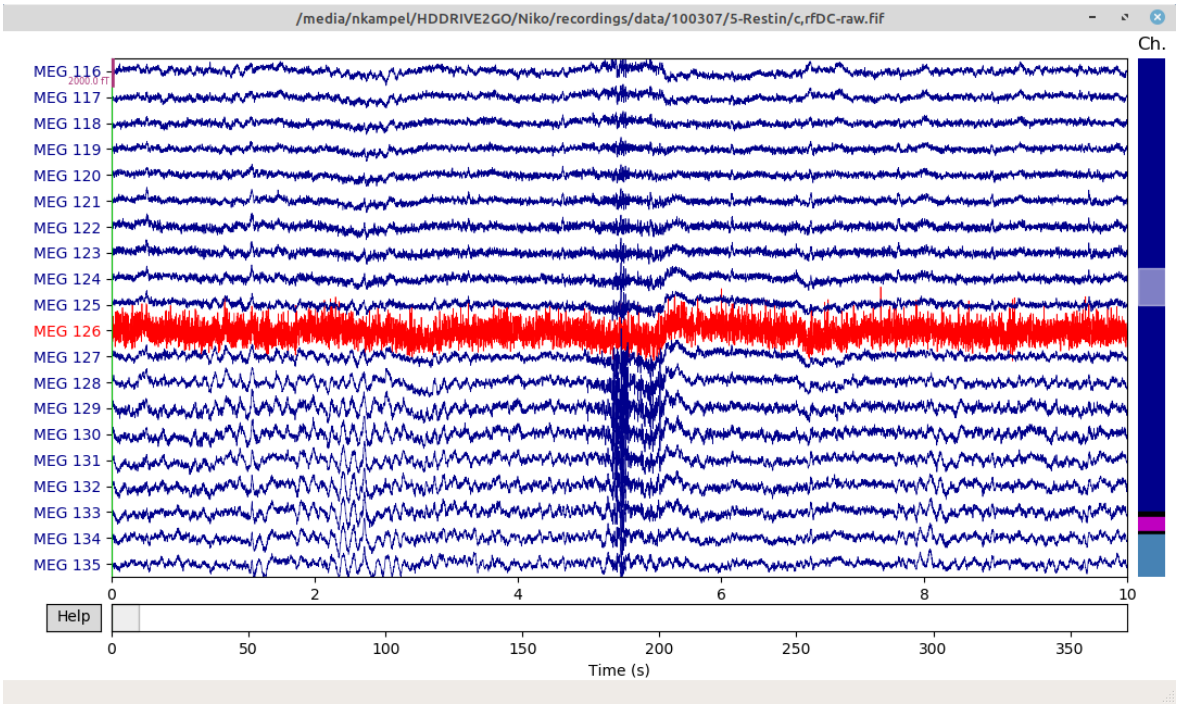

Figure 8. Increased noise (HCP Data)

2.2. Machine learning

This project uses machine learning algorithms in order to detect bad channels. There are many different types of machine learning algorithms and, therefore, it is very important to define exactly this term and what exactly it will mean for the project.

2.2.1. Definition

'Machine learning is an evolving branch of computational algorithms that are designed to emulate human intelligence by learning from the surrounding environment.' (4)

Despite popular belief, machine learning is not a synonym for artificial intelligence. The AI concept encompasses all those capabilities that a machine has to mimic human behavior. Machine learning is a section of artificial intelligence in which a machine imitates or tries to mimic the way humans learn.

On the other hand, another common confusion is to understand Deep learning and Machine Learning as synonyms. Deep learning is a part of machine learning that uses neural networks for model learning. It is the part of machine learning that most closely resembles human learning.

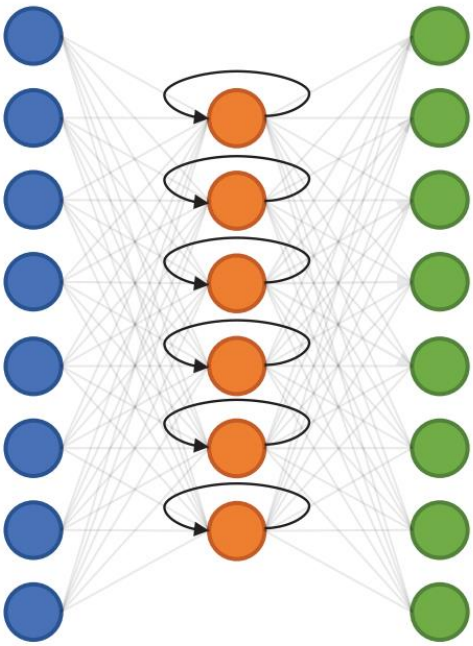

Figure 9. Recurrent neural network (5)

2.2.2. Application

The application of machine learning in this project will not use neural networks, so it will not be an application that uses deep learning. Instead, the proposals will use algorithms that, based on previously selected parameters, will be able to automatically classify and differentiate bad channels from good channels.

Finally, the reason why neural networks are not used is the scarcity of databases to train them. This fact makes the whole automation process very difficult.

2.3. MNE Package

MNE is the open source python package that has been used for the development of the project as it offers tools to explore and visualize different neurophysiological signals such as MEG and EEG among others.

The way the MNE package works is mainly with the use of objects. As mentioned above, the signals are stored in '.fif' files, an extension called Elekta Functional Imaging File Format widely used to store MEG and EEG files. Then, to read data so that it can be used in the multiple functions of the MNE

package, the 'read_raw_fif' function is used, which reads the signal and returns it as an object, so that the structure contains much more information than just the channels and recordings of the signal. It contains information such as the position of the electrodes, the distance between them, bad channels, annotations and more.(6)

2.3.1. Maxwell Approach

Within the MNE package there are certain functions to detect bad channels that complement the applications developed in the project. The latest and most interesting version is the maxwell bad channel detection method. It is very important for the understanding of the project to emphasize this function since at some points it will be taken as a reference and later an analysis of this function will be made using the developed analysis pipeline.

As any method currently available, it has strengths and weaknesses when applied to MEG signals. It is interesting to understand how the function works in order to try to create a complementary method that can make up for or cover the weaknesses of this one, so that the result can be a more robust pipeline with a higher percentage of success.

2.3.1.1. Maxwell Theoretical background

The function used is called 'find_bad_channels_maxwell'. First of all, we have to understand what it consists of.

Basically, the detection of bad channels of this method is based on using the SSS in the data to later compare the result with the original signal and to be able to detect outliers. But, what is this SSS? SSS stands for signal-space separation. It is used in multichannel signals to be able to separate the signal into two parts, which are called the outside and the inside. By inside it refers to all those signals that are related and come from the brain and its interactions and by outside it refers to all those interferences and artifacts present in the recordings that are caused by agents outside the brain activity, either by external magnetic fields or by artifacts from the sensors on the scalp.

As mentioned above, the result of applying SSS to a signal is a new signal from which all artifacts have been removed, also called 'device independent form'. Once the latter is available, first the channels with a low standard deviation are removed. Next, the difference in each channel between the original signal and the prediction is made. Then, the standard deviation of the differences and the mean are calculated and the z-score is calculated.

Once we have these values is when the role of the researcher comes into play. The function allows to set a threshold value from which the channels that exceed it will be marked as bad channels. An example of the result of the application of this method is the following:(7)(8)

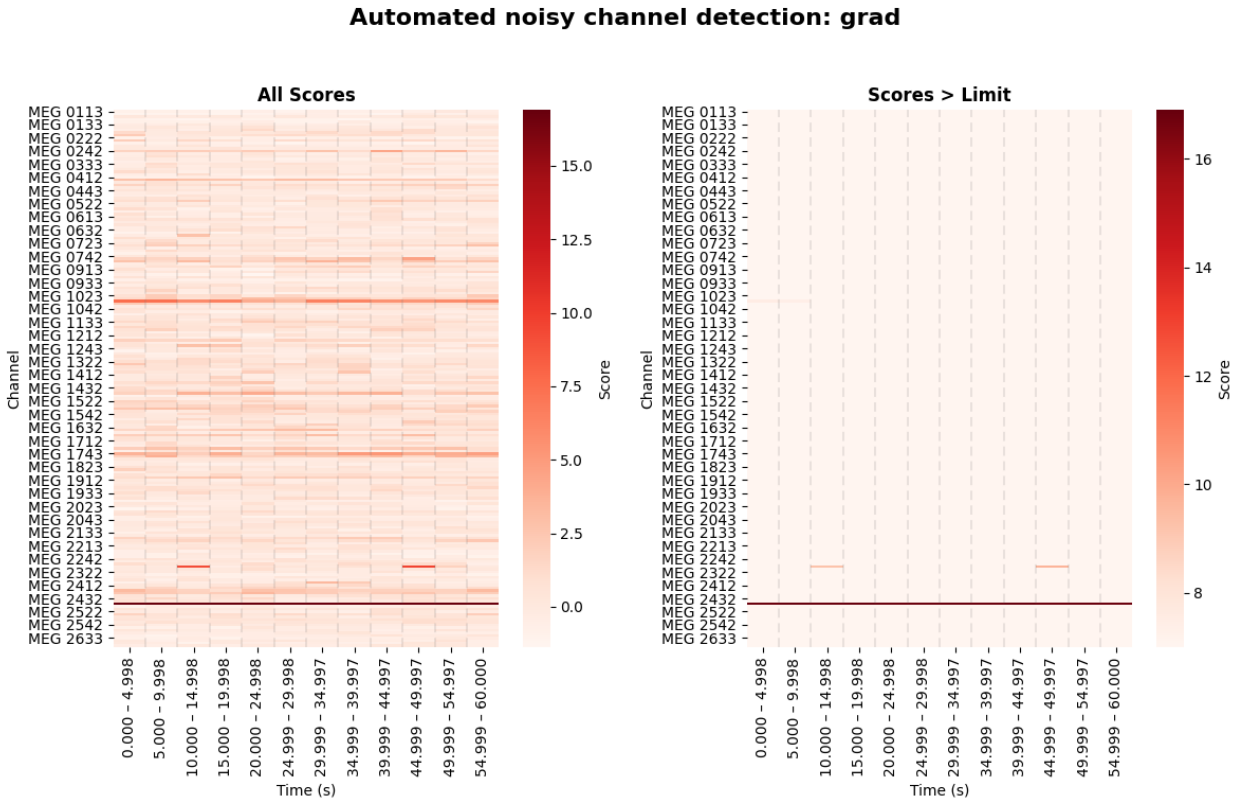

Figure 10. Maxwell results (9)

The first graph shows the z-scores of all the channels that have been calculated using the procedure explained above, and the second graph shows the result in the form of a graph after applying a threshold value to select the channels that are marked as bad channels, in this case channel 2432.(10)

2.3.1.2. Limitations

Although the method is robust in many circumstances it is far from perfect. There are certain limitations in applying this method that have to be considered when evaluating this method of bad channels detection:

- In order to work properly, it is necessary to know with precision the geometry of the sensors and their sensitivities. The problem with applying this method to the FZJ inhouse data is that in the .fif files it is not found in the reference channels, which causes the effectiveness of this method to be measurably reduced.

- The method has inconsistencies when detecting bad channels with the so-called 'flux jumps' disturbances. Since these are punctual errors in the channels and not a constant noise along them, the algorithm does not mark them as bad and therefore lets them pass. This implies that the method requires manual supervision.

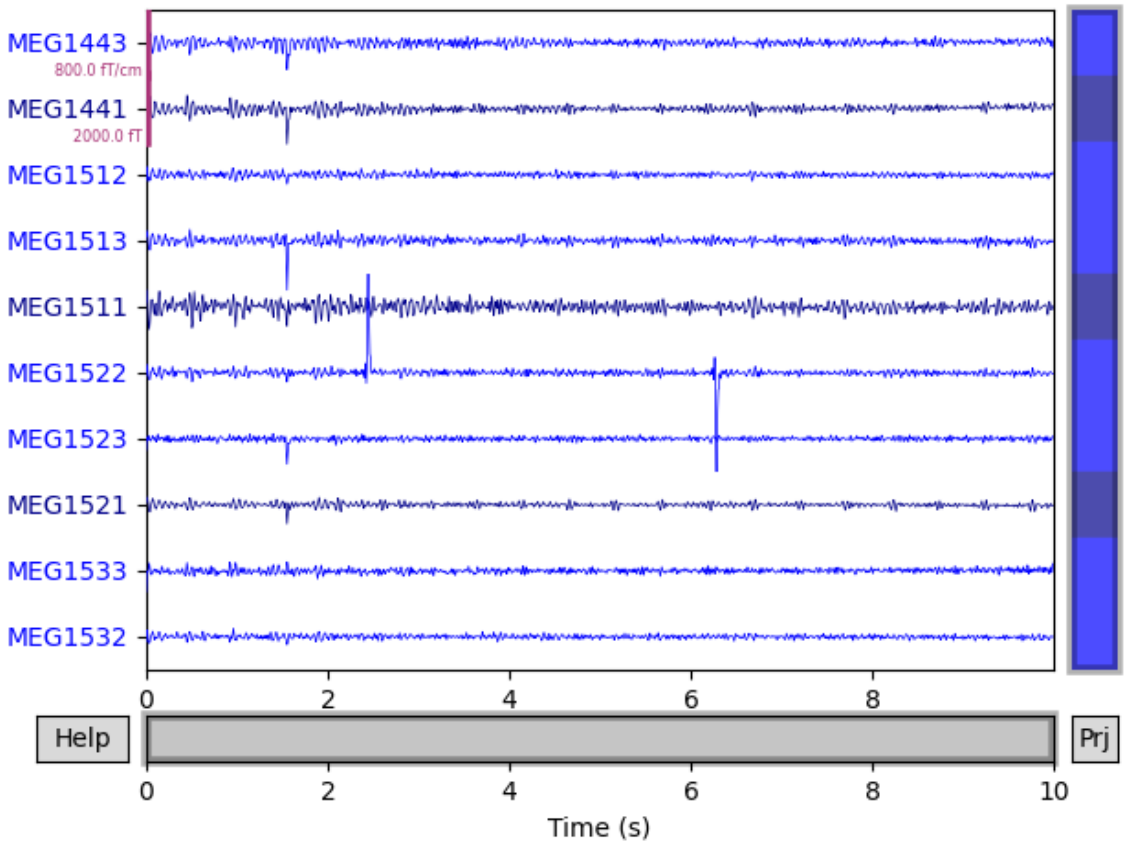

Figure 11. Flux jump on channel 1522 (11)

Note that this image shows examples of flux jumps on channel 1522, but it is not the same signal as in the image 9. (12)

2.4. HCP Data

HCP stands for Human Connectome Project. This is a project that aims to 'map the neural connections of the human brain'. It consists of several data sets collected over the last few years and for this project it has been worked with the signals from one of its latest projects, the WU-Minn HCP 500 Subjects+ MEG2 Data Release.

This release is accompanied by an extensively detailed manual, in which all the details of the data acquisition are explained. Among all this information there is the pipeline with which the HCP detects

bad channels. Being one of the most standard databases that can be found and having such a robust pipeline, the bad channels published by them have been taken as ground truth in the development of the project. The pipeline, far from being perfect, is currently the most consistent and rigorous in the preprocessing of MEG signals.

2.4.1. HCP Bad channel Pipeline

The pipeline is called 'baddata' and consists of several steps:(13)

- The first step is to detect bad channels by correlating each channel and its neighbors. A statistical threshold is used to determine which are the bad channels.
- Secondly, the Z score of all signal points is calculated and all points with a score greater than 20 are marked as bad.
- Then, excluding the channels already marked as bad, the standard deviation of each channel is calculated and compared with that of its neighbors. If the standard deviation of a channel is 50% greater than that of its neighbors, it is marked as bad.
- Finally, an Independent Component Analysis (ICA) method is used, which consists of separating the signal into different components, in order to identify artifacts that may be causing the bad channels.

The output of this pipeline is a text file in which all detected bad channels and segments are written.

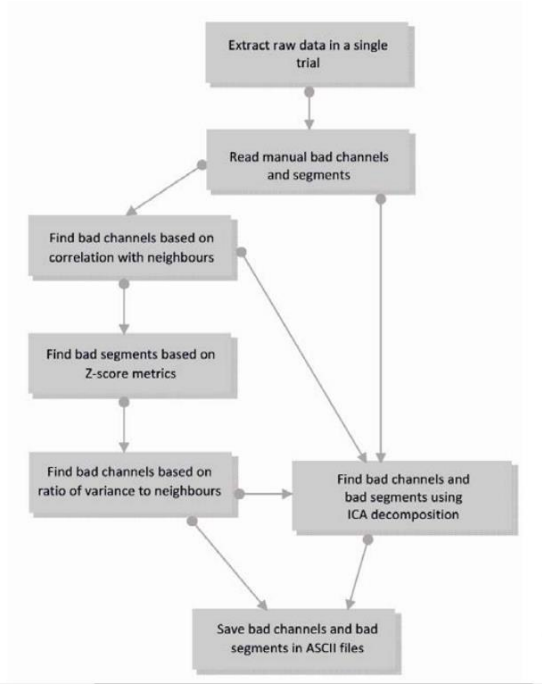

Figure 12. HCP baddata pipeline (3)

3. Methodology

This whole project was developed in the Python language, version 3.6. The environment used was the one directly provided in the MNE-Python page. (14)

The main objective of the methods used for the development of the project is that they are as standard as possible, so that they can be used in practically any type of signal, for that reason for its development has been used both a dataset of the FZJ and the previously mentioned dataset of the HCP. In this way the proposed methods are reproducible.

Note that the original HCP files are not directly in the .fif format but they are easily transformable to that format, so that at the time of starting this project, in the FZJ database this conversion was already done.

All the code can be found in the annex.

3.1. Bad channel detection

The first part of the project consists of two methods developed to detect bad channels in files with MEG signals.

3.1.1. Standard deviation

The first method developed was the calculation of the standard deviation. The objective is to make a heat map where outliers can be visually detected. This is achieved by first separating each channel into defined chunks called epochs. The criterion used to separate the epochs has been marked by the heartbeat. This ensures that the pieces of the signal are as similar as possible. For this purpose, the ECG signal was analyzed and an event was created for each heartbeat. The subsequent steps are rather simple, calculating the standard deviation at each epoch and the graphical representation of the result.

3.1.2. Neighbour correlation

The second method developed is the Neighbour correlation, called neigh_corr and located in the appendices. This method is based on the study (13), which demonstrates high coherence between signals that are close to each other and located in space.

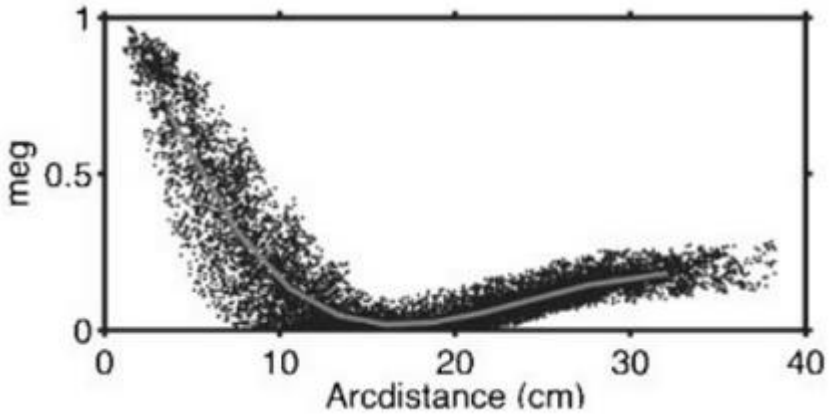

Figure 13. Coherence-distance relationship (13)

As previously mentioned, the .fif files contain the positions of each of the sensors. This information is very valuable when developing this method, since the first step is to calculate the distances between all the sensors from their positions. Once they have been obtained, 5 neighbors are found for each channel. This value has been selected as a suggestion, since it is the one that fits most of the signals, but for a more accurate analysis it is necessary to adapt this threshold to each database.

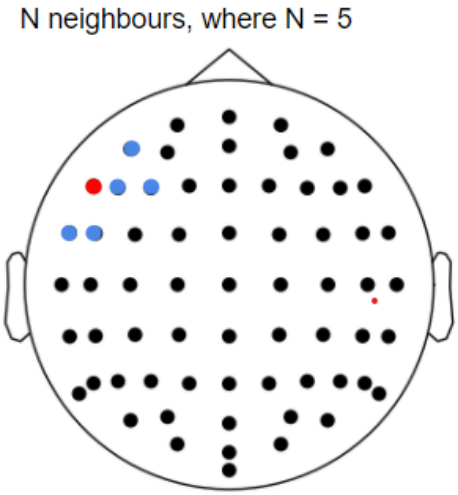

Figure 14. Neighbour correlation with 5 neighbours

Next, a matrix is created where the correlation values of each channel with its 5 nearest neighbors are stored. Once this matrix is created, the method used to detect bad channels is first to select the maximum of the 5 correlation values. Then, all those channels whose maximum correlation with their neighbors is less than 0.2 are marked as bad (as in the previous threshold, this is only indicative, since it varies a lot between different data sets). This has the advantage that, in the case that a channel is marked as bad, it has a very high probability of being correct since, for a channel to be bad, it has to have 5 very low correlations with all its neighbors. But on the other hand, this system presents a clear

problem and this is that a group of bad channels that correlate with high values will not be detected by the method.

Once the method was implemented, the results were not satisfactory. This is when an alternative to the neighbor system was proposed. The problem with determining a fixed number of neighbors for each channel is that the outer channels end up being correlated with channels farther away than the central channels. Therefore, the probability of poor correlation was higher for the outer channels.

To obtain an increase in accuracy, a new design for neighbor channel detection was considered. The premise of this design was, instead of using a number N of nearest neighbor channels as a boundary, to use a radius, so that depending on whether the sensor was located on the outside or inside, it would have a variable number of neighbors.

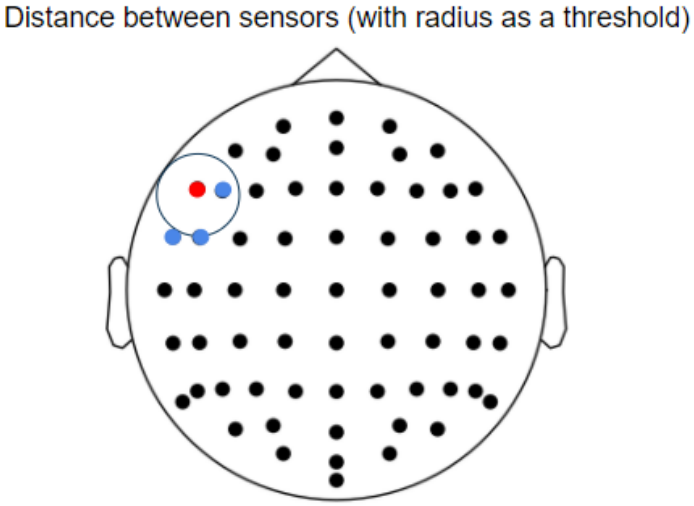

Figure 15. Neighbour correlation with radius as a threshold

3.2. Method analysis

Once the methods for detecting bad channels have been designed, the need arises for a method to test the effectiveness of the methods. In order to achieve this, the ' results' program is developed. What it does is to group in a dataframe the ground truth and predicted values (which are two arrays with the numbers of the bad channels in each file).

In this project, as mentioned above, the bad channels of the HCP are used as ground truth, because although they are not perfect, they are very good indicators as they have good accuracy.

The output of the results program is a list of files (one for each input signal) containing for each channel its prediction value and its ground truth value.

3.2.1. Confusion Matrix

Once you have that list of files, you simply have to apply a loop that collects all the data and then creates the confusion matrix. This is a tool that allows you to visualize the performance of the function that has performed the prediction. From this you can calculate many different parameters to understand the performance of the function, such as true positives and true negatives.

True positives are the number of times that a bad channel is marked as bad in the prediction and true negatives are the number of times that a good channel is labeled as good.

3.2.2. Bar plot

The matrix alone is not enough to accurately visualize the accuracy of the prediction. This is because, although the confusion matrix may show that the performance of the method is not good, with the bar plot it is possible to see which channels are the most marked as bad and, therefore, to see in which type of channels the two methods differ. This is extremely useful for correcting problems with the proposed method or to discover its strengths or weaknesses.

The development of this application is relatively simple, it simply requires adding up for each method the number of times a channel is counted as bad (i.e. across different files how many times, for example, channel 2 is counted as bad) and then creating a plot for each method. Finally, simply merge the two plots, obtaining the combined bar plot.

3.2.3. Final pipeline

Once the previous features have been obtained, the result is a pipeline that, from a list of .fif files, a ground truth and a method X that makes a prediction of bad channels, a confusion matrix is obtained with which the hits of bad channels detected as bad channels and the good channels detected as good can be evaluated, in addition to the bar plot to corroborate that the results of the confusion matrix have not been a coincidence and to observe which are the most problematic channels.

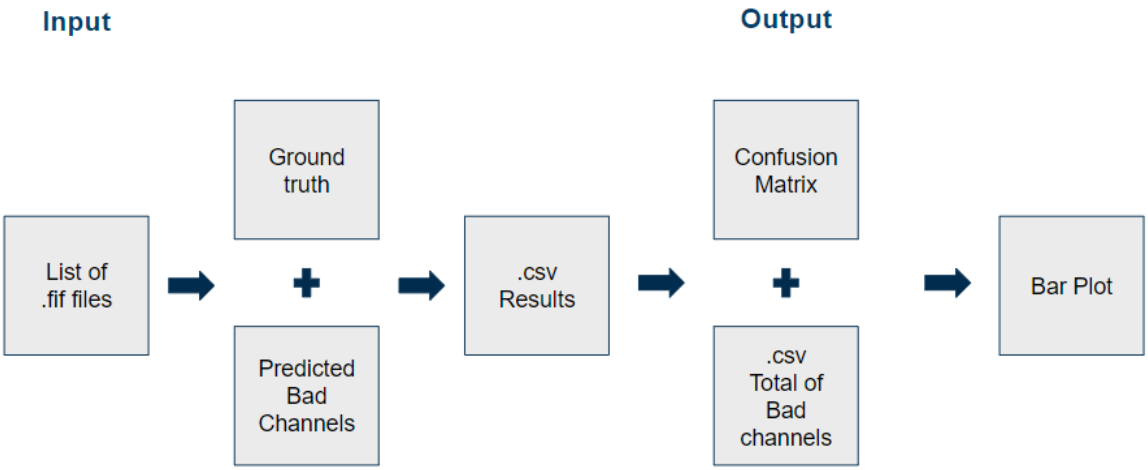

Figure 16. Final pipeline

4. Results and discussion

The following is an analysis of the results obtained from the methods developed above.

4.1. Bad channel detection

4.1.1. Standard deviation

The result of this application is a very useful heat map for the detection of outliers. The advantage of this method is that it is very visual, and therefore in case of a manual review it is extremely useful.

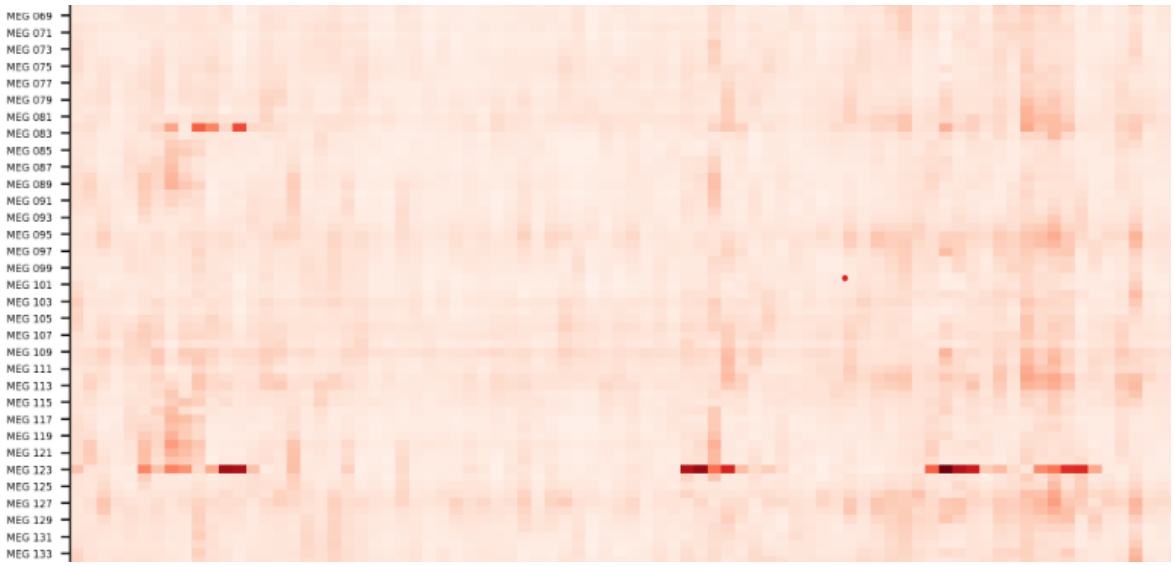

Figure 17. Standard deviation heat map

This is only a part of the total plot since the resulting plots are very large in size. It can be clearly seen how, in this particular example, channel 123 has several epochs where the standard deviation is very high, which probably implies that it is a bad channel. The downside of this method is the need for supervision for its effectiveness, which does not detract from the fact that it is extraordinarily practical.

4.1.2. Neighbour correlation

The change to selecting nearest neighbors by radius instead of by N number did not improve the results. This is because the premise of the method is that all neighbors must have a bad correlation to be detected as a bad channel, so further limiting the number of neighbors for the outer channels does not alter the result. The method also has a clear weakness and this is that in the case that instead of a single bad channel there is a group of bad channels, by depending so much on the reliability of the surrounding channels, the method loses all its effectiveness. Therefore, it can be concluded from this

method that it is a good intermediate step, since it can eliminate evident bad channels, but it requires further steps in order to be completely reliable.

4.2. Method analysis

The results of the performance of the developed pipeline have been carried out by testing the Maxwell bad channel detection method, described above.

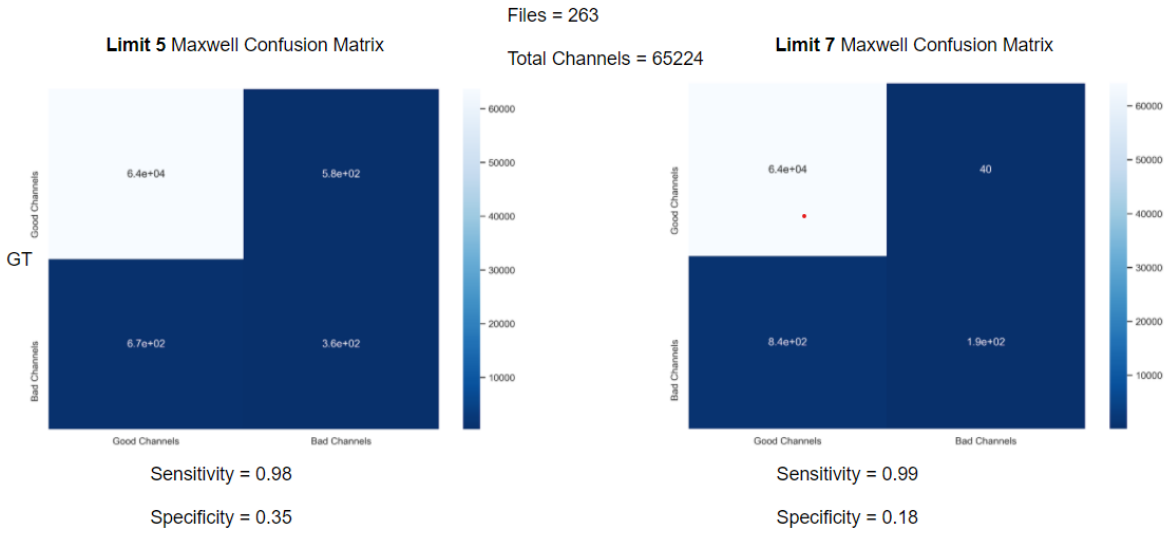

Figure 18. Confusion matrix of Maxwell with L 5 and 7

From these two confusion matrices the performance of the Maxwell method can be evaluated, but no extra information can be extracted. During the development of the project, just after performing the combined bar plot, in the HCP data we can observe a fact that simply with the confusion matrix could not be appreciated and that, obviously, altered the results. This is the fact that channel number 2 is always marked as bad channel. This fact could be due to many different reasons, such as the sensor being broken. But after further analysis, it could be seen that the problem with this channel was that it was inverted, and that caused the bad channel detector to mark it as bad. This fact corroborates the importance of having a complementary visualization to the confusion matrix, in order to perform a more accurate and reliable analysis.

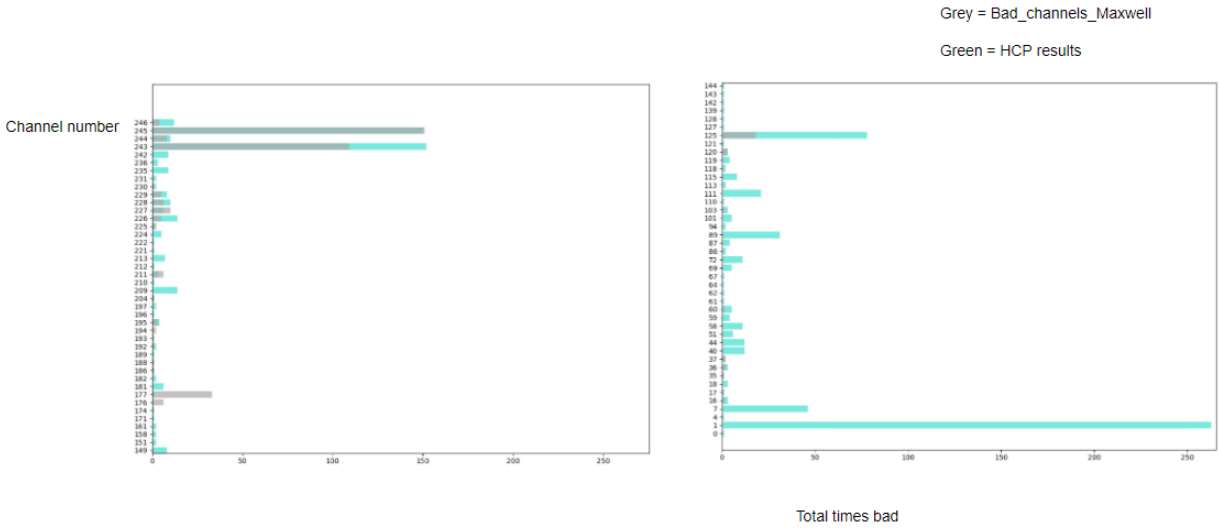

Figure 19. Bar plot of Maxwell (L=7) and HCP bads

4.3. Future improvements and development

First of all, the standard deviation method has a very clear possible improvement and that is to implement a system that, by using a threshold, is able to output a list of the channels with a higher deviation, thus avoiding the need to monitor the images manually in order to recognize the bad channels.

Secondly, it is worth mentioning that another method to detect bad channels was under development. This consisted in analyzing the PSD (Power Spectral Density) of the channels in order to extract features. More specifically, the procedure was as follows: Compute all PSD through a dataset and then comparing channels between each other (from different files).

In order to do that, the signal gets split into epochs, and then each epoch is compared among the different files (intersubject comparison). Channels that have bad segments representing over 25% of the whole channel are marked as bad. This method was not included in the project because it presented a major issue that was related to the wide range of results due to the file variation.

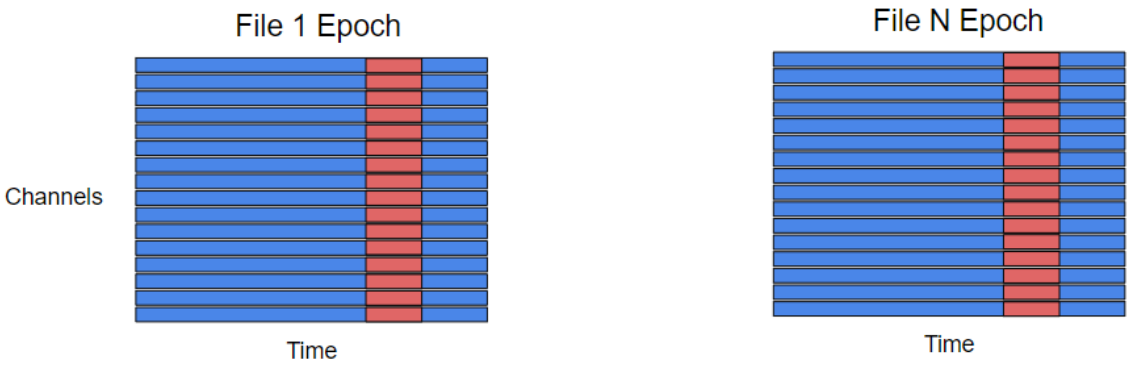

Figure 20. PSD epoch comparison across subjects

The last implementation involves using a split brain autoencoder. This consists of splitting the signal into two parts X1 and X2 and, using an autoencoder, making a prediction of X2 from X1 and vice versa. In this way, one could theoretically obtain a prediction of the signal that, when compared with the original, could be used to detect outliers in a robust way. The problem with these methods at present is that there is not enough data to be able to train a neural network completely.

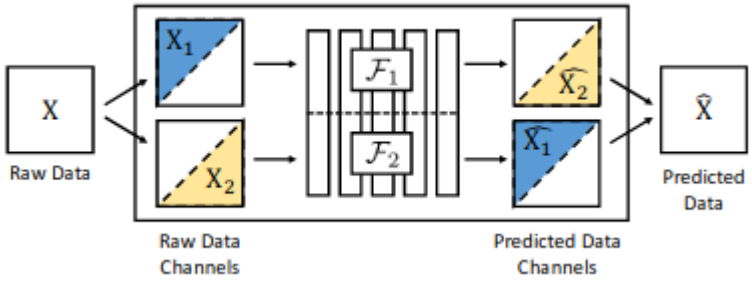

Figure 21. Split-brain autoencoder (15)

Conclusions

The field of bad channel detection is very broad and complex. Even so, the standard deviation and neighbor correlation methods are very useful despite their clear weaknesses. This shows that for the moment the solution is not a method that can detect all bad channels and is infallible, but that the best solution is a series of methods that complement each other to create a robust pipeline. The proposed methods are a small contribution to achieve this goal.

On the other hand, the analysis pipeline is very adaptable and flexible, so that it can be easily extended if extra features are needed. It is an efficient and fast way to evaluate the performance of any new method designed.

There are many possible improvements and future implementations, this is a growing field that has a great impact on the scientific community with the ability to improve the lives of many people.

Economic analysis

In this section the budget considering the development of the project will be presented; the cost of the personnel and materials used will be included in the budget.

Materials. This section will include the cost of materials that refer to the elements that were necessary for the development and implementation of the project, such as technological devices and programs. The following table shows the cost breakdown of the elements used:

Materials	Cost
Computer	1.700,00 €
Phyton	0 €
TOTAL COST	**1.700 €**

Table 1. Materials cost

Personnel. On the other hand, this section will consider the personnel costs. It should be clarified that these costs are based on subjective values considering that the development of the project consisted of tutoring meetings, research, methodology, results, conclusions, among others. The following tables show the cost breakdown of each activity carried out during the development of this project, considering the cost per hour. The cost per hour of the undergraduate student is 10 € and the cost per hour of the coordinator and co-coordinator of the project is 60 €.

- Student cost.

Activity	Cost per hour	Number of hours	Total Cost
Planning	10 €	40	400,00 €
Investigation	10 €	60	600,00 €
Proyect Development	10 €	600	6.000, 00 €
Document Drafting	10 €	80	800, 00 €
TOTAL COST			**7.800,00 €**

Table 2. Student cost

- Coordinator and co-coordinator cost.

	Activity	Cost per hour	Number of hours	Total Cost
COORDINATOR	Tutoring and Project coordination	60 €	100	6.000,00 €
CO- COORDINATOR	Tutoring	60 €	50	3.000,00 €
TOTAL COST				**9.000,00 €**

Table 3. Coordinator and co-coordinator cost

Total cost of the project. The following table shows the total cost of the project considering the cost of materials and the cost of personnel.

Cost type	Cost
Materials	1.700,00 €
Degree student	7.800,00 €
Project coordinator and co- coordinator	9.000,00 €
TOTAL COST	**18.500,00 €**

Table 4. Total cost

Environmental impact analysis

In order to develop the project, different technological devices such as the computer were necessary. As is well known, this type of increasingly effective technologies are very valuable as tools for analysis and management of programs used for the collection, pre-processing, processing and diagnosis of brain signals, among other applications it may have. However, it should be noted that this type of technology has a negative side, synonymous with environmental impact. In order to reduce this impact in the development of the project, some measures were taken into account:

- Turn off computers and other devices that are not in use.
- Connect the computer for the time necessary to charge the battery.
- Adjust the computer to work in an energy saving mode.
- Work in daylight environments to avoid the use of artificial light.

Bibliography

1. Hansen, P., Kringelbach, M. i Salmelin, R. MEG: An Introduction to Methods. In: 2010

2. J Vrba, SE Robinson, J.M. How many channels are needed for MEG? - PubMed. In: *2004 Nov 30, 2004:99* [online]. [Accesed: January 2021]. Available: https://pubmed.ncbi.nlm.nih.gov/16012656/.

3. WU-Minn HCP 500 Subjects + MEG2 Data Release: Reference Manual. In: . 2014.

4. El Naqa, I. i Murphy, M.J. What Is Machine Learning? A: *Machine Learning in Radiation Oncology* [online]. Springer International Publishing, 2015, p. 3-11. DOI 10.1007/978-3-319-18305-3_1. [online]. [Accesed: February 2021]. Available: https://link.springer.com/chapter/10.1007/978-3-319-18305-3_1.

5. Litjens, G. et al. A Survey on Deep Learning in Medical Image Analysis. In: 2017

6. Seven stories about MNE — MNE 0.14.1 documentation. [online]. [Accesed: February 2021]. Available: https://mne.tools/0.14/tutorials/seven_stories_about_mne.html.

7. Taulu, S. i Kajola, M. Presentation of electromagnetic multichannel data: The signal space separation method. A: *Journal of Applied Physics* [online]. American Institute of PhysicsAIP, 2005, Vol. 97, num. 12, p. 124905. ISSN 00218979. DOI 10.1063/1.1935742. [Accesed: March 2021]. Available: http://aip.scitation.org/doi/10.1063/1.1935742.

8. Taulu, S. i Simola, J. Spatiotemporal signal space separation method for rejecting nearby interference in MEG measurements. A: *Physics in Medicine and Biology* [online]. IOP Publishing, 2006, Vol. 51, num. 7, p. 1759-1768. ISSN 00319155. DOI 10.1088/0031-9155/51/7/008. [Accesed: February 2021]. Available: https://iopscience.iop.org/article/10.1088/0031-9155/51/7/008.

9. Signal-space separation (SSS) and Maxwell filtering — MNE 0.23.0 documentation. A: [online]. [Accesed: January 2021]. Available : https://mne.tools/stable/auto_tutorials/preprocessing/60_maxwell_filtering_sss.html#taulukajola2005.

10. mne.preprocessing.find_bad_channels_maxwell — MNE 0.23.0 documentation. A: [online]. [Accesed: February 2021]. Available: https://mne.tools/stable/generated/mne.preprocessing.find_bad_channels_maxwell.html.

11. sphx_glr_otp_001.png (640×480). A: [online]. [Accesed: March 2021]. Available in: https://mne.tools/stable/_images/sphx_glr_otp_001.png.

12. Taulu, S. i Kajola, M. Presentation of electromagnetic multichannel data: The signal space separation method. A: *Journal of Applied Physics*. 2005, Vol. 97, num. 12. ISSN 00218979. DOI 10.1063/1.1935742.

13. Winter, W.R. et al. Comparison of the effect of volume conduction on EEG coherence with the effect of field spread on MEG coherence. A: *Statistics in Medicine* [online]. John Wiley & Sons, Ltd, 2007, Vol. 26, num. 21, p. 3946-3957. ISSN 02776715. DOI 10.1002/sim.2978. [Accesed: March 2021]. Available: http://doi.wiley.com/10.1002/sim.2978.

14. Installing MNE-Python — MNE 0.24.dev0 documentation. A: [online]. [Accesed: March 2021]. Available: https://mne.tools/dev/install/mne_python.html.

15. Zhang, R., Isola, P. i Efros, A.A. Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction. A: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* [online]. Institute of Electrical and Electronics Engineers Inc., 2016, Vol. 2017-January, p. 645-654. [Accesed: March 2021]. Available : http://arxiv.org/abs/1611.09842.

Annex

A1. Standard deviation plot (std_plot_map)

```
import os

from copy import deepcopy

import numpy as np

import mne

import glob

import matplotlib

import matplotlib.pyplot as plt

def std_plot_map (raw):

#Only take the MEG channels

picks = mne.pick_types(raw.info,meg=True,include='bads')

#Detecting the events using the EEG peaks

events = mne.preprocessing.find_ecg_events(raw)

#Setting an epoch window using the average pulse and calculating ECG epochs

window= 60/events[2]

ecg_epochs = mne.preprocessing.create_ecg_epochs(raw,picks=picks,tmin=-window/2,

 tmax=window/2, reject_by_annotation=False)

#Setting the times when each of the events occurs

eventtime=[]

for i in events[0][:,0]:
```

```
 eventtime.append(raw.times[i])


roundedtime=np.around(eventtime,1)

#Calculating the std of each epoch

epoch=np.array(ecg_epochs)

std= np.std(epoch, axis=2)

#Getting the name of each channel

n=len(picks)

channelnames= raw.ch_names[0:n]

#Getting parameters for the plot

m=len(std[:,0])

unodecadados=channelnames[::2]

p=len(unodecadados)
oneoftwo=std[::3]

q=len(oneoftwo)

#Name of the file

basefolder = '/home/avillalba/Desktop/Images/'

parts = file.rsplit('/')[-1].rsplit(',')[0].rsplit('_')

image_name = parts[0]+'_'+ parts[1] + '.png'

fileout = basefolder+image_name

#Plotting the std

a=123*4.2/p

b=40*p/123
```

```
c=10*q/72

std1= np.transpose(std)

matplotlib.rc('ytick', labelsize=a)

matplotlib.rc('xtick', labelsize=a)

fig, ax= plt.subplots(figsize=(b,c))

locs, labels = plt.yticks()

plt.xticks(range(q),roundedtime)

plt.xticks(np.linspace(0, m, q))

plt.yticks(range(p) ,unodecadados)

plt.yticks(np.linspace(0, n, p))

ax.imshow(std1,cmap='Reds')

fig.colorbar(ax.imshow(std1,cmap='Reds', aspect='auto',interpolation='nearest'))

plt.savefig(fileout, bbox_inches='tight',dpi=300)

plt.clf()
```

A2. Neighbour correlation (Neigh_corr)

```
import os

import mne

import glob

import numpy as np

import matplotlib.pyplot as plt

from scipy.spatial import distance

from mpl_toolkits.mplot3d import Axes3D
```

```
def _distance_from_channels(pos_4d, channel_x, channel_y):
 # given sensor location and 2 channel indices, compute their eucledian distance
 loc_x, loc_y = pos_4d[channel_x] , pos_4d[channel_y]
 ch_distance = np.abs(distance.euclidean(loc_x,loc_y))
 return ch_distance


def compute_distance_matrix(raw):
 # select the
 picks = mne.pick_types(raw.info, meg='mag',exclude=[])


 chs = [raw.info['chs'][pick] for pick in picks]
 pos_4d = np.array([ch['loc'][:3] for ch in chs]) # look up table of sensor coordinates location given in
meters
distance_matrix = np.zeros((len(pos_4d),len(pos_4d))) # initialise matrix with zeros
 for i1, channel_x in enumerate(pos_4d): # loop through x indices
 for i2, channel_y in enumerate(pos_4d): # loop through y indices
 distance_matrix[i1,i2]=_distance_from_channels(pos_4d, i1, i2)
 return distance_matrix, pos_4d
def get_nearest_neigbours(selected_channel,distance_matrix, num_of_neigbours = 5):
 neighbours = (dm[selected_channel]).argsort()[1:num_of_neigbours+1]
 return neighbours
def neigh_corr (raw):
```

```
dm, pos = compute_distance_matrix(raw_data)

picks = mne.pick_types(raw.info, meg=True)

correlationmatrix=[]

for x in picks:

correlationvalues=[]

neigh = get_nearest_neigbours(x, dm, 5)


for y in neigh:

r = np.corrcoef(raw._data[x,:,], raw._data[y,:,])

correlationvalues.append(r[0,1])


correlationmatrix.append(correlationvalues)


a = np.array(correlationmatrix)

maxlist = []

z = -1

for row in a:

z = z+1

b = np.max(a[z])

maxlist.append(b)

maxlistnp = np.array(maxlist)

corrbadchannels = []
```

```
 for w in maxlistnp:
 if w < 0.2:
 corrbadchannels.append(maxlist.index(w))
corrbadchannels = np.array(corrbadchannels) + 1
 return corrbadchannels
```

A3. Results pipeline

```
#%%
from tqdm import tqdm
import os
from copy import deepcopy
import numpy as np
import mne
import glob
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
from datetime import datetime
#%%
filelist = glob.glob('/home/avillalba/Desktop/HCP/hcp/*.fif')

date = datetime.now().strftime('%d%m%Y_%H%M%S')
basefolder = '/home/avillalba/Desktop/HCPTest/'
```

```
comment = '' + '_Maxwell_V1_L4'

folder_name = 'r_'+ date + comment

directory = basefolder + folder_name

os.makedirs(directory)

chosen_method = myfunction

#%%



for file in tqdm(filelist):


    raw = mne.io.read_raw_fif(file,preload=True)


        #Extracts the already bad channels of the file as a ground truth

    hcpgroundtruth1 = (raw.info['bads'])


    hcpgroundtruth = []


    #Converts them to integers

    for val in hcpgroundtruth1:

        hcpgroundtruth.append(val[-3:])


    hcpgroundtruth = [int(a) for a in hcpgroundtruth]
```

```
#Creates 0 array

groundtruth = [0] * 248

groundtruth = np.array(groundtruth)


#Substitutes 0 for 1 taking the bad channels (-1) as an index

for idx, val in enumerate(groundtruth):

   for y in hcpgroundtruth:

      if y == idx:

       groundtruth[idx-1] = 1


raw.info['bads'] = []

raw._data[248:,] = 0

print('!!!!!!!!!!')

alp = chosen_method(raw)

#alp, olp = mne.preprocessing.find_bad_channels_maxwell(raw,limit=4.5,coord_frame = 'meg')

hcppredicted = []

#Converts them to integers

for val in alp:

   hcppredicted.append(val[-3:])


hcppredicted = [int(a) for a in hcppredicted]
```

```
#Creates 0 array

predicted = [0] * 248

predicted = np.array(predicted)



#Substitutes 0 for 1 in every bad channel calculated by the method

for idx, val in enumerate(predicted):

   for y in hcppredicted:

      if y == idx:

       predicted[idx-1] = 1



#Corr Bad Channels

# corrbadchannels =   neigh_corr(raw)

#Substitutes 0 for 1 in every bad channel calculated by the method

# for idx, val in enumerate(predicted):

#     for y in corrbadchannels:

#       if y == idx:

#        predicted[idx-1] = 1



#Creating a dataframe from both np stacked

groundandpred = np.matrix.transpose(np.vstack((groundtruth,predicted)))
```

```
groundandpred = pd.DataFrame({'Ground Truth':groundandpred[:,0],'Predicted Bads':groundandpred[:,1]})


#Save the file

basefolder = '/home/avillalba/Desktop/HCPTest/'+ folder_name +'/'

parts = file.rsplit('/')[-1].rsplit(',')[0].rsplit('-')[0]

file_name = 'r_'+ parts + '.csv'

fileout = basefolder+file_name

groundandpred.to_csv(fileout)
```

A4. Confusion matrix

```
import seaborn as sn

import os

from copy import deepcopy

import numpy as np

import mne

import glob

import matplotlib

import matplotlib.pyplot as plt

import pandas as pd

from datetime import datetime

from sklearn.metrics import confusion_matrix

from sklearn.metrics import plot_confusion_matrix
```

```
from sklearn import svm

from sklearn.metrics import ConfusionMatrixDisplay as cmd

plt.ioff()

#%%

#Loads most recent folder

filelist = glob.glob (sorted(glob.glob('/home/avillalba/Desktop/HCPTest/*'),key=os.path.getmtime)[-1]+'/r_*')

#Load selected folder

#filelist = glob.glob('/home/avillalba/Desktop/HCPTest/r_24092020_173926/*.csv')

totalofbads = []

cmdata = []

for file in filelist:

    a = 0

    data = pd.read_csv(file)

    cmdata.append(data)

    a = len(data[data['Predicted Bads'] == 1])

    totalofbads.append(a)



cmdata = pd.concat(cmdata)

cmdata = cmdata.rename(columns={'Unnamed: 0': 'Channel Number'})
```

```
cm = confusion_matrix(cmdata['Ground Truth'],cmdata['Predicted Bads'])

labels = ['Good Channels', 'Bad Channels']

df_cm = pd.DataFrame(cm, index = labels,

            columns = labels)

plt.figure(figsize = (10,7))

sn.heatmap(df_cm,cmap='Blues_r', annot=True)

fileout = file.split('r')[0] + file.split('/')[5] + '/' + 'ConfusionMatrix'+ '.png'

plt.savefig(fileout, bbox_inches='tight',dpi=300)
```

```
totalofbads = pd.DataFrame(totalofbads)

#Use this if using different thresholds (e.g:maxwell_V1_L6)

#totalofbads = totalofbads.rename(columns={0: 'N of Bads per file' + ' ' + file.rsplit('/')[5].rsplit('_')[5]})

#Use this to name it

totalofbads = totalofbads.rename(columns={0: 'N of Bads per file' + 'correlation' })

fileout = file.split('r')[0] + file.split('/')[5] + '/' + 'nbads' + '.csv'

totalofbads.to_csv(fileout)
```

A5. Bar plot

```
import glob

from tqdm import tqdm

import os
```

```
from copy import deepcopy

import numpy as np

import mne

import glob

import matplotlib

import matplotlib.pyplot as plt

import pandas as pd

filelistmaxwelll8 = glob.glob('/home/avillalba/Desktop/HCPTest/*L7/r*')

file = pd.read_csv(filelistmaxwelll8[0])

bads_list = np.array([0]*248)

gt_bads_list = np.array([0]*248)

#Stack up bads by channel and

#select second row of each csv file and attatch them, summary of each row

for file in filelistmaxwelll8:

    q = pd.read_csv(file)

    file_values = []

    file_values = q.values.tolist()

    file_values = np.array(file_values)

    bads_list = [bads_list[i]+file_values[i][2] for i in range (len(bads_list))]

    gt_bads_list = [gt_bads_list[i]+file_values[i][1] for i in range (len(bads_list))]



gt_bads_list = np.array(gt_bads_list)
```

```
bads_list = np.array(bads_list)

bads_list = np.c_[bads_list, file_values[:,0]] #attaching the channel numbers to the bad channels


barplotdf = pd.DataFrame(data = bads_list, columns = ['Total Bads','Channel Number'])

barplotdf['GT Total Bads'] = gt_bads_list


#%%

brplotdf = barplotdf.drop(columns=['Channel Number','Total Bads'])


#%%

brplotdf1 = barplotdf.drop(columns=['Channel Number', 'GT Total Bads'])

#%% Plot taking out the channels where both are 0

barplotdf2 = barplotdf.loc[(barplotdf['Total Bads']>0) | (barplotdf['GT Total Bads']>0)]

barplotdf2['Channel Number'] = barplotdf2 ['Channel Number'].astype(str)

plt.bar(barplotdf2['Channel Number'],barplotdf2['GT Total Bads'],alpha = 0.7, color = 'turquoise')


plt.bar(barplotdf2['Channel Number'],barplotdf2['Total Bads'], alpha = 0.7, color = 'darkgrey')


plt.show()
```