Building and Interpretation of a Seizure Classifier using Graph Signal Processing

From Electroencephalographic Data

Raphaël Mariétan

EPFL Signal Processing Laboratory

Semester Project Supervised by Arun Venkitaraman

January 7, 2021

Table of Contents

1 Introduction

Epilepsy is one of the most common neurological disorder affecting about 1% of the population, and consists in the recurrent expression of unprovoked seizures. A seizure occurs when there is an alteration of neurological functions due to an imbalance between excitation and inhibition in the brain, resulting into excessive synchronous discharge of neurons in the brain. Epilepsy is a condition that has been affecting the lives of many people, but the neurological mechanisms at the origin and occurring during a seizure are still not fairly uncovered. Due to the lack of understanding and to the high variety in the expression of this neurological condition, there is still about a third of the epileptic patients that do not receive appropriate treatment (Stafstrom and Carmant, 2015). Indeed, seizures can be classified into many different types and their expression can be quite varied, resulting in treatments having variable efficiency depending on the condition of each patient. Appropriate diagnosis is therefore essential to provide the right treatment to a patient.

Visual analysis of EEG scans by a specialist remains to this day the most common way to categorise the type of condition and to identify the right treatment plan. This diagnosis method is time-consuming and highly prone to variations in inter-human abilities to identify the right condition. With the help of accurate computational model, it could become easier and faster to classify seizures from EEG recordings. Fitting a model to the available data could also be useful to discover more about the hidden neurological pathways that encode the mechanisms of a seizure.

The first goal of the current project is to learn a meaningful representation of the connectivity of the brain when it is experiencing a seizure, through the process of learning a connectivity graph from the 20-channel EEG recordings that are provided in the dataset described in the next section. The next step will be to try to use these graph representations of the seizure recordings as input for the training of a seizure type classifier. This graph classification can ultimately be used as a tool to try to interpret which features, and therefore which connections were the most responsible for the classification of each seizure type. This last step will be performed by running an explainability analysis on the classifier that was trained on the graphs. The whole implemented pipeline and more technical information about its implementation can be found on the following repository : https://github.com/Rmarieta/LTS4.

2 Dataset

The dataset used for the current project is taken from a large database of electroencephalographic recordings conducted at Temple University Hospital (TUH) from 2002 on. Having large amounts of data in a consensual format is essential in order to provide deep learning models with consistent and significant material to improve with. The advances in the domain of neuroscientific domain are highly relying on the availability of reliable and extended data, whether it is to validate models or uncover new mechanisms by building new models. This project is using the latest version (to this day) of the NEDC TUH EEG Seizure dataset, v1.5.2, released in 2020 and available on TUH EEG Corpus.

This dataset gathers EEG recordings from many patients undergoing various types of seizure events across multiple sessions. The EEG recorded signals are stored in a standard European Data Format (EDF) and these EDF files contain the recordings from multiple electrodes placed according to the 10-20 EEG placement system. The figure below (Figure 1) depicts the organisation of the dataset, which is split into a train and test dataset, each containing many patients (identified by a number e.g. 00592) and many sessions (e.g. s001) of recordings (e.g. t001.edf).

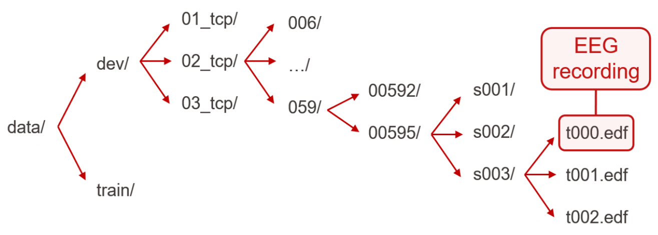

Figure 1: Diagram of the tree architecture representing how the EEG recordings are stored in the dataset.

In addition to the EDF files, a binary output of the temporal occurrence of the seizure events is provided for each sample, as well the type of seizure that the patient was undergoing. By keeping all the recordings in which a seizure event took place, the train dataset contains a total of 2377 seizures and the dev dataset a total of 673 events, adding up to a total of 3050 seizures available for the development of the model. The seizures are labelled per type and their distribution is detailed in Table 1.

Seizure Type	Patients	Seizures
Focal Non-Specific (FNSZ)	150	1836
Generalised Non-Specific (GNSZ)	81	583
Complex Partial (CPSZ)	41	367
Absence (ABSZ)	12	99
Tonic (TNSZ)	3	62
Simple Partial (SPSZ)	3	52
Tonic Clonic (TCSZ)	14	48
Myoclonic (MCSZ)	3	2

Table 1: Distribution of the seizure counts per seizure type in the recordings.

Considering the reduced amount of data available for all seizure types except FNSZ and GNSZ, only these two types are considered for the current model. As explained later, the variability in the graphs generated for a type are such that it is necessary to have a substantial amount of data for that seizure type in order to learn specific features.

3 Methods

In this section, the theory behind each step of the implementation is explained, starting from dealing with the data until trying to explain what features of each seizure type of interest, namely FNSZ and GNSZ, are relevant to their classification.

3.1 Data Extraction

The first step of any model dealing with signals is always to extract the part of the signals that are targeted, before saving it into the right format for the following steps. The current implementation of the data extraction part is an adaptation of the work described in (Asif et al., 2020), and their code can be found on Github. In addition to the EDF files containing the signals, four other files are required for the data extraction, and in particular for the seizure extraction :

- data_preparation/parameters.csv, which contains the sampling frequency, as well as the channel settings used to save signal into an array of 20 channels. There is a wide range of reference montages to be used in clinical EEG studies (Acharya et al., 2016). The channel mapping used here is shown on Figure 2.

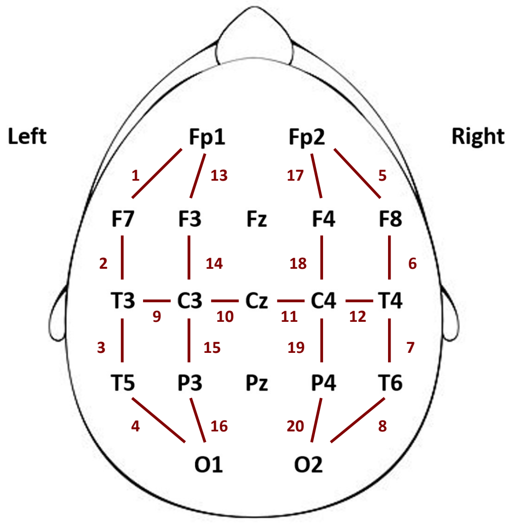

Figure 2: Channel mapping used in the project, e.g. here channel F7-T3 will correspond to the second entry of the 20-channel extracted signal.

- data/v1.5.2/_DOCS/seizures_v36r.xlsx, which is the file used to retrieve all the seizure times and their type. This excel file contains two excel sheets, one for train/ and one for dev/, and lists all the EDF file names as well as the start and end times of a seizure, in case one occurred during the recording. Iteration through this excel file allows to eventually build distinct folders for each seizure type only containing sections of recordings with seizure events. Each seizure is stored under the form of a pickle file.

- data/v1.5.2/_DOCS/ref_dev.txt and data/v1.5.2/_DOCS/ref_dev.txt, from which it is possible to extract the binary output for each recording of when the patient is undergoing a seizure and when the patient is resting. This is useful when the extraction of the background samples is needed. It can then be treated as a class (BG) with each pickle file containing the section of a recording when a patient is not undergoing a seizure (amongst the recordings that are labelled as containing at least one seizure event).

3.2 Graph Computation

The next step of the implementation is to go from these time-series representation of the seizure events to a graph representation in order to hopefully learn some interesting results from the signals that are more interpretable regarding the underlying brain spatial connectivity. Using 3 different algorithms, we can generate a [20x20] matrix whose entries are computed from the signals :

- A first possible way to get such a graph is to compute the covariance matrix on the 20-channel signal, which outputs a [20x20] symmetric matrix that can be used as a graph for later classification.

- The other two algorithms used here are attempting at learning the adjacency matrix from the signals, hence the topology of the underlying graph, under the assumption that the signals are smooth. The first of these two algorithms is learning the Laplacian matrix from a multi-channel signal, and is an implementation of the framework described in (Dong et al., 2016). The adjacency matrix is a matrix whose element$_{i,j}$ indicates whether the pair of channels made of channel$_i$ and channel$_j$ are adjacent in the graph representation. A high value would suggest that these two channels are highly connected according to the graph it represents. It is therefore the representation of the connectivity that we wish to learn under the form of a matrix. The adjacency matrix A can be easily retrieved from the Laplacian L computed in the current algorithm using the following formula, with D_L the diagonal matrix constructed with the diagonal of L :

$$A = -(L - D_L) \tag{1}$$

  The implementation of this algorithm can be found on the following Github.

- The last technique tried in the project is an adaptation of the framework that learns the graph structure underlying a set of smooth signals detailed in (Kalofolias, 2016). The implementation of this technique can be found on the following repository. This algorithm directly learns the adjacency matrix from the set of 20 signals that we have from each recording.

Using the 3 techniques described above, we can generate a graph for any of our multi-channel signals. We then try two different ideas for the graph computation step. We can either generate one graph per seizure event, or generate one graph per second of seizure recording in order to have a higher time resolution and maybe less smoothing out of some important events, but a lot more variability on the other hand.

3.3 Classification

Once the graphs are computed, we train a classifier that aims at classifying each matrix as either FNSZ or GNSZ. The different classifiers proposed here can also be used with more than two classes, under some slight adaptations in the size of the hidden and output layers of the neural networks, as well as for the oversampling of the under-represented classes, which will have to be checked when adding another class. Due to the strong differences in sample sizes (especially if we try with the background class, as so many samples are available), the classes are highly imbalanced. To deal with that, we randomly duplicate the necessary amount of graphs from the under-represented class to make it balanced. The detailed implementation of these classifiers can be found in the following repository.

3.3.1 Simple models

The first option available is to train some simple models implemented with the *scikit-learn* library. The models implemented here are the Bayes Classifier, k-Nearest Neighbor, Support Vector Machines, Classification Tree and Multinomial Logistic Regression. The model is trained using the seizures extracted from train/, and tested on the data from dev/. The classifier is trained on the flattened upper triangle of the matrix, as the matrices are symmetric no matter the technique used in the graph computation step.

3.3.2 Feed-forward Neural Network

In addition to the fairly simple models described above, a standard feed-forward neural network is implemented using *PyTorch*. The input of the neural network is also flattened here and only the upper triangular part of the original matrix is kept as a 1D input. The neural network takes the input, feeds it through 3 linear layers with max pooling, ReLU and drop-out and outputs either FNSZ or GNSZ as a class.

3.3.3 Convolutional Neural Network

In this second neural network, the input graphs are considered as 2D images of 20x20 pixels. This neural network is also tried on the Laplacian matrices in addition to the adjacency matrices to see if we get different performance. The lower triangular part of the matrix can be set to 0 or not, as the matrix is once again symmetric. The input goes through 2 convolutions and 3 linear layers before outputting the class.

3.4 Graph Explainability

The model learned for the classification, in particular in the case of a neural network, acts as a black box that outputs a prediction for the class, but we do not have direct access to what features of the FNSZ graphs triggered the network to classify them as FNSZ. The most interesting part of the problem is obviously to be able to understand what characterises a focal seizure as compared to a general seizure, and what changes in what channels could be responsible for the expression of a focal or a general seizure. Answering such extrapolating questions through the interpretation of the trained network is a very complex task, but there exist some libraries such as SHAP, which aim at providing different tools to explain the output of any machine learning model. The DeepExplainer functionality can help here with the explainability as we can use our graphs as 2D images and try to see, for each class, which pixels play the most recurrent role into determining the output. As detailed in the Results section, SHAP can be used to compute a mask per class of positive and negative entries for any image, named SHAP values. For instance, a positive SHAP value in a pixel of the mask of the FNSZ class means that in the particular case of this input graph, this pixel pushed the prediction higher into classifying the image as FNSZ. By running this DeepExplainer on the model trained for the classification, and averaging the SHAP masks computed on all the graphs, we can hopefully see what pixels seem to have the most important role into classifying the computed graphs the right way, knowing their ground-truth classification.

4 Results

4.1 Graph Computation

Using the 3 techniques mentioned in the Graph Computation Methods section, we get upper triangular matrices of variable appearance, which are displayed in this section. To start with, the algorithm from (Kalofolias, 2016), which learns an adjacency matrix from the signal, surprisingly yields fully-connected matrices, and was therefore discarded for the classification.

For the remaining two algorithms, graphs were computed with and without low-pass filtering the signals with multiple cut-off frequencies (30, 50 and 100Hz), so as to see if it has any influence on the classification accuracy. Using covariance matrices as graphs yielded graphs which show large connectivity. Figure 3 and 4 respectively depict the output graphs computed on the FNSZ and the GNSZ seizure samples. It is hard to notice any pattern proper to each class solely based on the visual appearance of the graphs for each class.

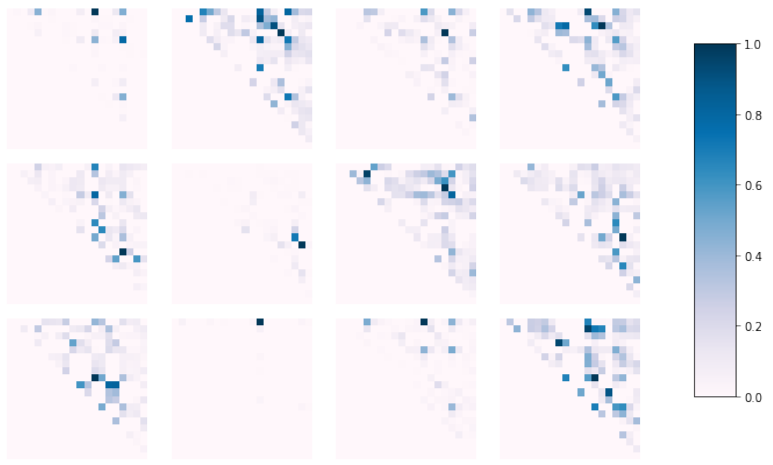

Figure 3: Samples of graphs that were computed by taking the covariance matrix of the FNSZ signals.

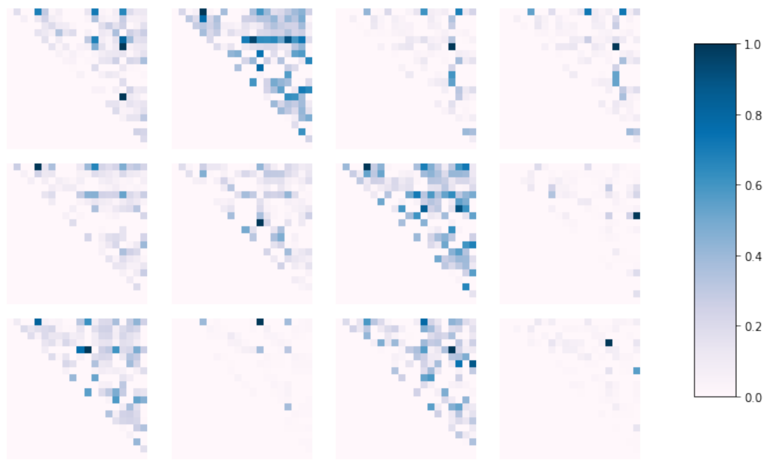

Figure 4: Samples of graphs that were computed by taking the covariance matrix of the GNSZ signals.

Using the adjacency matrix learned in the algorithm described in (Dong et al., 2016), we get the output plotted on Figure 5 and 6, respectively for the FNSZ and GNSZ signals. We notice that as seen on Figure 5, some matrices show surprising patterns of over-connectivity with a cross of zero entries, while the remaining graphs seem to display features that are distinct in the two classes. The samples that unexpectedly resulted in over-connected graphs were discarded for the later classification.

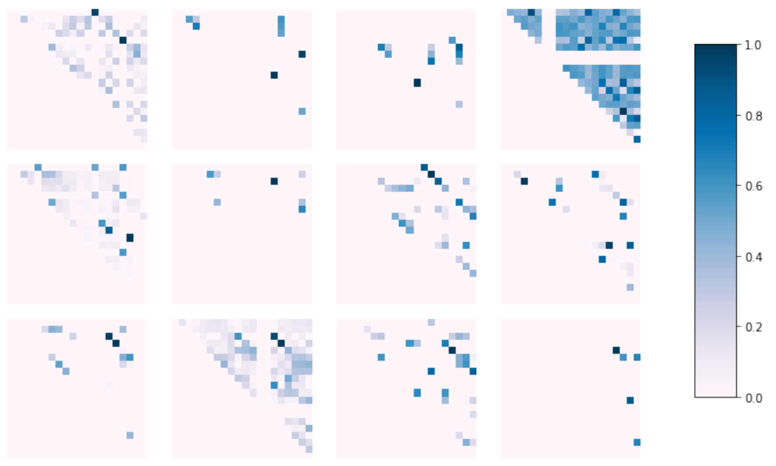

Figure 5: Samples of graphs that were computed by learning the adjacency matrix of the FNSZ signals with the algorithm from (Dong et al., 2016).

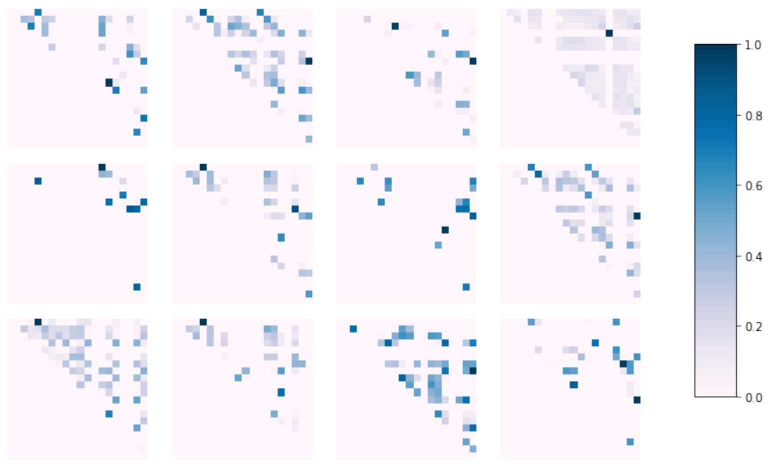

Figure 6: Samples of graphs that were computed by learning the adjacency matrix of the GNSZ signals with the algorithm from (Dong et al., 2016).

All the graphs on Figures 3, 4, 5 and 6 were computed by previously low-pass filtering the signals with a cut-off frequency of 50Hz and by computing one graph per seizure event. When computing the graphs on 1 second chops of recordings, we get a total of 56510 graphs for FNSZ and 25943 graphs for the GNSZ class (before balancing the classes), as compared to 1522 (FNSZ) and 408 (GNSZ) graphs computed on full recordings. We then have many more samples for the classification but the graphs display a lot more variability than the already highly-variable full-recording graphs due to the short length of the signals.

4.2 Classification

For the classification task, multiple combinations of graphs and classifiers were used as an attempt to reach the highest accuracy on the testing set, as well as a balanced per-class testing accuracy. That is, we prefer a model that outputs 65% testing accuracy for both classes over another model that outputs 80% accuracy for FNSZ and 50% accuracy on GNSZ, despite the fact that it would yield the same F1-score on balanced testing sets. Tables 2 and 3 show the results of some of the attempts made using train/ as the training set and dev/ as the testing set for the models detailed on the left. The combination outputting the best result occurs when combining the two graph computation methods, covariance and adjacency matrix, and using both to form a 2-channel input for the CNN described in Convolutional Neural Network section, and we name this new network as dual CNN.

Classifier	Input Graph	FNSZ Acc.	GNSZ Acc.	Weighted F1-score
kNN	Adjacency	60.0%	44.0%	53.3%
	Covariance	69.0%	44.0%	60.0%
	Adj. (1s chops)	66.0%	32.0%	61.6%
	Cov. (1s chops)	70.0%	37.0%	61.4%
SVM	Adjacency	83.0%	28.0%	59.2%
	Covariance	85.0%	33.0%	63.5%
	Adj. (1s chops)	89.0%	12.0%	68.8%
	Cov. (1s chops)	91.0%	21.0%	66.8%
Bayes	Adjacency	53.0%	69.0%	59.6%
	Covariance	85.0%	41.0%	67.2%
	Adj. (1s chops)	76.0%	19.0%	63.7%
	Cov. (1s chops)	78.0%	18.0%	58.7%
Logistic Regression	Adjacency	65.0%	57.0%	62.4%
	Covariance	58.0%	65.0%	61.5%
	Adj. (1s chops)	69.0%	32.0%	63.2%
	Cov. (1s chops)	80.0%	51.0%	72.0%

Table 2: Accuracy on test set and weighted F1-scores of the simple models selected on the left and using the graphs mentioned in the 2^{nd} column as input.

Table 2 shows the per-class accuracy and the weighted F1-scores with some simple classifiers. We see that they overall have a low F1-score and that they are often biased towards outputting one class, except with logistic regression. In addition, we see that computing graphs on 1 second chops of the signals changes the F1-score, but it highly modifies the balance between the per-class accuracies, the classifiers then tend to lean towards classifying all samples as either FNSZ or GNSZ. This is highly undesirable as the classification becomes less accurate than flipping a coin for one class. The 1 second chops with the covariance matrix behaves surprisingly well with the logistic regression but is once again biased towards FNSZ.

Table 3 below shows the outputs of the neural networks. Here, the accuracy with and without low-pass filtering and with and without chopping the signals into 1 second long signals is shown in the case of the first model. As the low-pass filtering step slightly improves the accuracy, and as once again, the 1 second chops result in a unwanted high bias towards one class, we only display the best combination for the remaining two networks.

Classifier	Input Graph	FNSZ Acc.	GNSZ Acc.	Weighted F1-score
FFNN	Adjacency	61.5%	60.8%	61.9%
	Low-pass, Adj.	63.9%	59.2%	62.6%
	Adj. (1s chops)	76.4%	25.2%	65.8%
	Covariance	63.3%	64.9%	64.5%
	Low-pass, Cov.	65.0%	64.4%	65.3%
	Cov. (1s chops)	79.9%	42.0%	68.7%
CNN	Low-pass, Adj.	70.8%	67.2%	69.8%
	Low-pass, Cov.	71.0%	50.6%	63.5%
Dual CNN	Low-pass, Adj. + Cov.	80.2%	65.5%	74.6%

Table 3: Accuracy on test set and weighted F1-scores of the simple models selected on the left and using the graphs mentioned in the 2nd column as input. FFNN : Feed-Forward Neural Network.

4.3 Graph Explainability

After trying multiple models in combination with different graphs, we obtain a weighted F1-score of 74.6% in the case of taking both the covariance and the adjacency matrices as input for our dual CNN. The confusion matrix of this model is displayed below on Table 4. The testing accuracy achieved is 80.2% for the FNSZ class and 65.5% for GNSZ.

		Predicted label	
		FNSZ	GNSZ
True label	FNSZ	231	57
	GNSZ	60	114

Table 4: Confusion matrix of the selected model.

As explained in Graph Computation, the SHAP library provides tools for attempting to give a meaningful interpretation of the classifier output relatively to the spatial connectivity. Using the DeepExplainer tool, it is possible to generate a [20x20] mask of SHAP values for each graph and in combination with each class. Figure 7 is an example of the output of the DeepExplainer computed on two test graphs, one FNSZ and one GNSZ graph. It always outputs two classes here as it shows the SHAP values when explaining the output class FNSZ and the other SHAP values when explaining the class GNSZ. A red, positive SHAP value means that the feature associated with the positive-valued pixel is pushing the prediction towards the explained class higher. A blue pixel means that the feature is pushing the prediction away from the explained class.

Here, we see that different pixels played a role in explaining the output for each input graph, and if we compute the SHAP masks for each input graph that we have, we can try to sum them all up into one single graph per class to see if some specific pixels, hence some specific channels, stand out in the explanation of the output class. Figure 8 below shows the output that we obtain after thresholding the resulting summed up contributions of all SHAP masks. The pixels with high value in the FNSZ graph below mean that these pixels were more recurrently involved in rightly pushing the prediction of the output FNSZ higher (as only the SHAP masks on test graphs that were rightly classified by the trained model were taken into account).

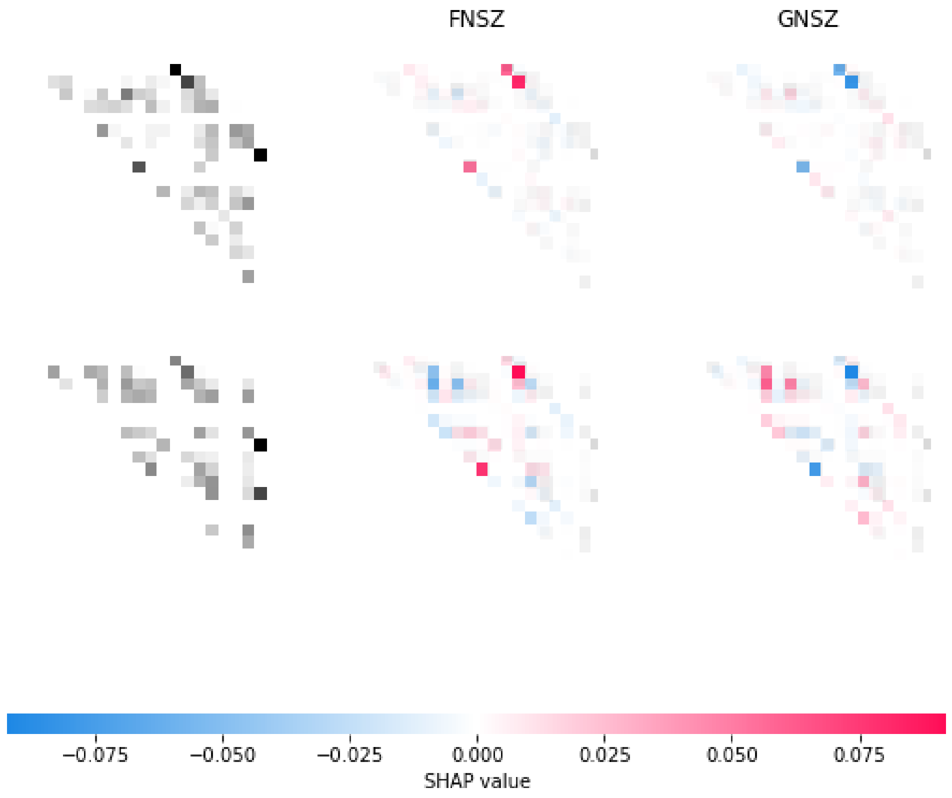

Figure 7: Two examples of SHAP masks computed on two test graphs. The left column represents the two input graphs (top : FNSZ, bottom : GNSZ for true class). The 2nd column shows the SHAP values when the output is FNSZ, and the 3rd column shows the mask relative to GNSZ as the classifier output.

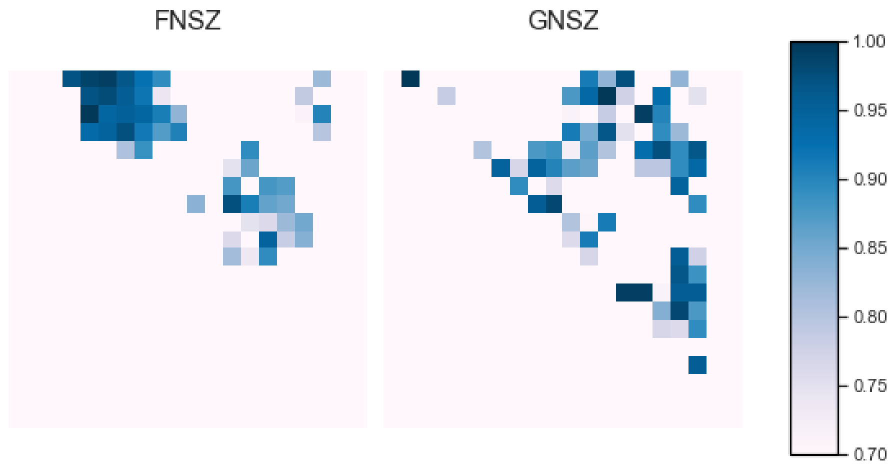

Figure 8: One graph per class summing up the contributions of each pixel to the output class by computing the SHAP masks on all available test graphs.

5 Discussion

Concerning the computation of the graphs, we visually notice that there is a lot of variability in the patterns displayed in the graphs of each seizure type, which we could expect, but for different patients of the same class as well. This high variability in the overall shape of all signals results in the inability to obtain a consistent graph representation for the background parts of the recordings (parts with no seizure event). It is also hard to tell whether the brain connectivity of a patient is already altered before the occurrence of a seizure or still altered after the event, or even continuously altered as being a subject affected by the neurological condition. Having a control would be important in order to draw more reliable conclusions on what characterises each seizure type condition as compared to the brain at rest.

With the Laplacian learning technique, some graphs are extremely sparse, while some show high connectivity despite the samples representing the same type of seizures. The learning algorithm might not be behaving expectingly due to the complexity of the signals. Indeed, this learning algorithm is expected to learn a valid Laplacian matrix from the signals assuming that those signals are smooth, a condition that might be hard to meet when dealing with EEG recordings. On top of that, the high inter-personal characteristics of EEG recordings make it hard for an algorithm to learn a consistent connectivity pattern, as the recordings from different patients will show significant differences which are not necessarily related to the expression of epileptic condition. Having as many patients and as many recordings as possible is therefore essential to build a robust representation that does not fits to the samples it is being trained with. Overall, some distinct features in the connectivity patterns of the different seizure types can still be highlighted from the graph representation, whether it is obtained using the covariance matrix or the adjacency matrix algorithm, as reflected during the classification.

The issue of inter-patient differences also translates into the difficulty for all the classification attempts to yield consistent accuracy. The neural networks were subject to strong overfitting, which had to be counterbalanced with a strong drop-out, to avoid scaling the model to the characteristics of the patients whose recordings are used for the training. Most of the models tested in this project tended to over-classify and scale to the characteristics of one class, which is not desirable, as we want the testing accuracy to be at least higher than 50% for all classes. The selected model is able to rightly classify almost 3 quarters of the test samples, which made it relevant to run a feature explanation on the trained model.

Using the DeepExplainer tool, a graph summing up the contributions of the pixels of each graph towards the explanation of the output class was obtained and displayed on Figure 8. Looking at the two graphs, we see that different areas of pixels seem to be more often involved in determining the class for both seizure types. In particular, we can clearly see a cluster forming on the top left of the FNSZ graph, which might suggest that the first channels play a more recurrent role into deciding which class will be attributed to a graph. According to the channel mapping detailed on Figure 2, the particular order in which the channels are ordered and used as index for the graph relates to some form of spatial proximity on the scale of the brain. Indeed, two neighboring pixels on the graph translate into channels of related areas in the brain. The areas involving the channels that are deemed more relevant to the classification according to our model explanation are highlighted on Figure 9 below. These channels were obtained by retrieving which channels were the most recurrent across the SHAP masks positive values.

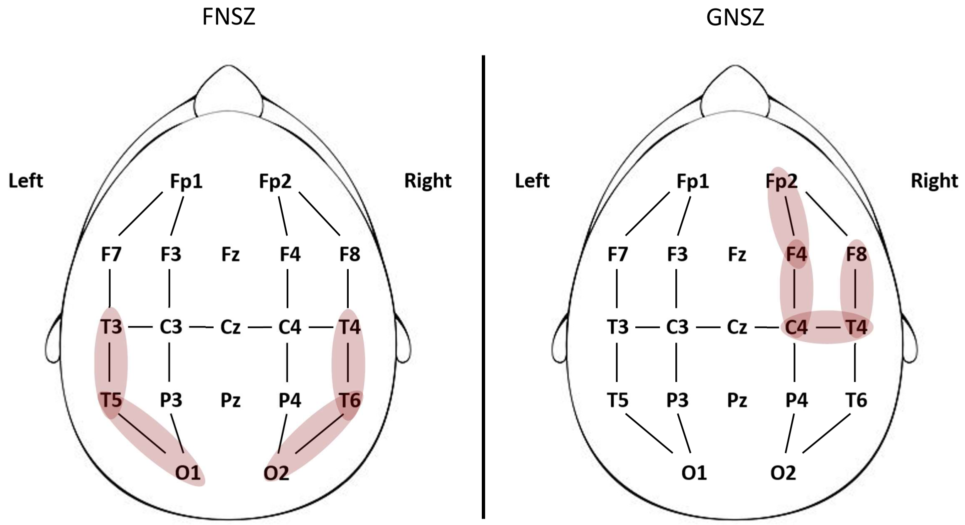

Figure 9: Most occurring channels for each class when running the SHAP explainability analysis.

Although the pattern of highlighted channels seems distinct between FNSZ and GNSZ, caution needs to be taken on the interpretation of these results. The explainability analysis run here is not a way to deduce what areas or channels can be held accountable for the expression of the FNSZ or the GNSZ condition, as it is attempting to explain the difference between the two classes and what makes the train classifier pick one or the other. Coming back to the cluster on the top left of Figure 8, it might be that it is rather the absence of some pattern in these top left pixels in the FNSZ graphs that helped the classification of such graphs as FNSZ rather than GNSZ. We can not decipher to which class the explainability can be attributed to, as in the current development, only differences can be discussed. Even though the conclusions that we can draw from the explainability analysis are restricted, it still builds an interesting parallelism between the spatial connectivity of the brain and some synchronised time series, and also gives an insight on the important regions related with epileptic condition.

To follow up on the current project, it would make sense to try to find a way to have a valid graph representation of what a background sample is; that is having a meaningful representation of a graph built on a signal displaying no seizure event. This way, it would be more valid to draw conclusions on what channels are linked with the expression of a seizure thanks to having a control class. This would allow to do more than observing the difference between seizure types. Adding more seizure types in the classification task could also be interesting, but the lack of a statistically significant amount of data for the classes other than FNSZ and GNSZ makes it hard with the current data to overcome the high variability of the signals and therefore to run a consistent model explanation once the classifier is trained.

6 References

Acharya, J. N., Hani, A. J., Thirumala, P., and Tsuchida, T. N. (2016). American Clinical Neurophysiology Society Guideline 3: A Proposal for Standard Montages to Be Used in Clinical EEG. *The Neurodiagnostic Journal*, 56(4):253–260.

Asif, U., Roy, S., Tang, J., and Harrer, S. (2020). SeizureNet: Multi-Spectral Deep Feature Learning for Seizure Type Classification. *arXiv:1903.03232 [cs, q-bio, stat]*. arXiv: 1903.03232.

Dong, X., Thanou, D., Frossard, P., and Vandergheynst, P. (2016). Learning Laplacian Matrix in Smooth Graph Signal Representations. *IEEE Transactions on Signal Processing*, 64(23):6160–6173.

IBM (2020). Seizure type classification - tuh. `https://github.com/IBM/seizure-type-classification-tuh`.

Kalofolias, V. (2016). How to learn a graph from smooth signals. *arXiv:1601.02513 [physics, stat]*. arXiv: 1601.02513.

Mariétan, R. (2022). Lts4. `https://github.com/Rmarieta/LTS4`.

Rodrigo, C. G. P. (2017). Smooth graph learning. `https://github.com/rodrigo-pena/graph-learning`.

Slundberg (2018). Shap. `https://github.com/slundberg/shap`.

Stafstrom, C. E. and Carmant, L. (2015). Seizures and Epilepsy: An Overview for Neuroscientists. *Cold Spring Harbor Perspectives in Medicine*, 5(6):a022426. Number: 6.

TheShadow29 (2018). Learn graph laplacian. `https://github.com/TheShadow29/Learn-Graph-Laplacian`.