Detection of areas affected by Epilepsy based on EEGs through deep learning

Alejandro Gutierrez Gouverneur [1], Ana María Maitín [2]

[1] Departamento de Ciencias de la Computación, Universidad de Alcalá de Henares; 28801 Alcalá de Henares, Madrid, Spain; alejandroenrique.gut@edu.uah.es (A.G.G.)
[2] Centro de Estudios e Innovación en Gestión del Conocimiento (CEIEC), Universidad Francisco de Vitoria, 28223 Pozuelo de Alarcón, Spain; a.maitin@ceiec.es (A.M.M.)

Abstract: Recently, machine learning techniques have been introduced in biomedicine, specifically epilepsy classification models using electroencephalograms (EEGs) distinguishing abnormal scalp EEGs of patients which were originally classified as 'normal' by experts. On the other hand, given the spatial coverage of the EEG which is not a regular grid we've considered graphs as the structure for our neural networks, that's Graph Convolutional Networks. We're using EEGraph to model EEGs as graphs, so the connectivity between different brain areas could be analyzed and StellarGraph for deep learning on graphs. Our experiments will be modeled with a publicly available large scalp from Physionet EEG Siena-EEG with 14 subjects.

Keywords: Epilepsy; electroencephalography (EEG); machine learning (ML) Graph Neural Networks (GCN)

1. Introduction

Epilepsy is interpreted as a neurological disorder characterized by the occurrences of seizures, due to abnormally excessive or synchronous neuronal activity in the brain. Very importantly, the main features of epilepsy, epileptic seizures, are associated with several negative consequences at both short- and long-term, including the risk of falls and injuries, psychiatric disturbances, cognitive deficits, difficulties in achieving academic, social and employment goals, and eventually death. Treatment options for epilepsy are pharmacological and surgical.

However, antiepileptic drugs have limitations and fail to control seizures in roughly 30% of patients, while surgery is not always an option. In this context, an important issue is the possibility of predicting/detecting the occurrences of epileptic seizures (i.e., detecting a preictal or pre-seizure state) in order to take actions to neutralize an incoming seizure or limit the injuries (e.g., by warning alarms, applying short-acting drugs, activating stimulating devices).

To carry out the study, we will use the electroencephalograms (EEGs) of patients with epilepsy that experience seizures during recording of the EEG. Electroencephalography is a clinical test that is responsible for recording the electric field produced by pyramidal neurons in the brain, resulting in records of brain activity over time. An EEG consists of several electrodes distributed over the head of the patient, so that each of these electrodes will provide a time series that contains the information of the brain activity of

a specific region of the brain. Among the properties of this technique, it is worth mentioning that it is a non-invasive technique for the patient, of rapid execution, high reproducibility, high temporal resolution and low cost in comparison with other tests commonly used in the field of neurology. However, these advantages are eclipsed by the characteristics of the EEG signals, these are, stochasticity and high noise level. Conventional EEG analysis techniques involve artifact cleaning and noise level minimization processes, along with complex statistical analyzes to obtain the desired information. However, Deep Learning techniques have proven to be powerful enough to be able to bypass the preprocessing of EEG signals and obtain results with high precision and efficiency.

In this project we will use Graph Convolutional Neural Networks (GCNN or GCN), a new technique to obtain results in graph structured data. This model allows classification problems at graph level and at node level. Therefore, in a first step, the binary classification problem (Preictal-interictal) will be covered, to later rethink the model at node (electrode) level and look for the characteristic patterns of the disease different areas of the brain. For this, the EEGraph Python library will be used, which oversees modeling the EEGs as graphs, which will be then introduced to the model trough Stellargraph which as is a Python library for machine learning on graphs and networks.

2. Related Work

Epileptic seizure prediction literature with EEG has experienced fast-paced development in the most recent years, they're mainly based on two different approaches: seizure detection (which aims to the classification of seizure-non seizure) and seizure prediction (that aims to the classification for preictal and interictal stages mainly, although sometimes it adds a third class for ictal stage).

Tsiouris et al. [1] extracted descriptive features from EEG and obtained a 99.8% accuracy on the CHB-MIT-EEG dataset using a Long-Short-Term-Memory (LSTM) network. Chen et al. [2] showed an EEG Graph Convolutional Network (E-GCN) method for classification and detection of epilepsy that obtains an accuracy of 98.4% classifying ictal and interictal periods on the CHB-MIT-EEG dataset. Zeng et al. [3] proposed a model that makes a full utilization of the locational topological relationship in electrodes on the scalp and treats different adjacent relationship by Hierarchical Graph Convolutional Networks (HGCN) obtaining a 99.9% accuracy in the prediction of interictal and preictal stages. Dissanayake et al. [4] proposed a Geometric Deep Learning (GDL) technique for epileptic seizure prediction which applied a naive graph computation approach using the physical structure of the EEG composition obtaining a 95.4% on its subject independent distance-based approach. Li et al. [5] propose a Spatio-Temporal-Spectral Hierarchical Graph Convolutional Network with Active Preictal Interval Learning (STS-HGCN-AL) that captures temporal dynamics in an epileptic cortex under different rhythms and that obtains an AUC of 0.94.

Our work is then inspired by my recent advances in the field of neural networks that operate on graphs. For our GCN we've decided to make use of use the spectral graph convolution propagation rule defined by Kipf and Welling [6] which at the same time is a first-order approximation of spectral convolutions defined by Defferrard [7]. To achieve fast and efficient local filtering, local filter was obtained using the Chebyshev polynomial [7], and it was applicable to any graph structure.

3. Data available

The database consists of EEG recordings of 14 patients acquired at the Unit of Neurology and Neurophysiology of the University of Siena [8-9]. Subjects include 9 males (ages 25-71) and 5 females (ages 20-58), they were monitored with a 29 channel (after discarding EKG, SPO2, HR) Video-EEG with a sampling rate of 512 Hz, with electrodes arranged based on the international 10-20 System as it can be observed in Figure 1. The data were acquired employing EB Neuro and Natus Quantum LTM amplifiers, and reusable silver/gold cup electrodes. Patients were asked to stay in the bed as much as possible, either asleep or awake.

The diagnosis of epilepsy and the classification of seizures according to the criteria of the International League Against Epilepsy were performed by an expert clinician after a careful review of the clinical and electrophysiological data of each patient.

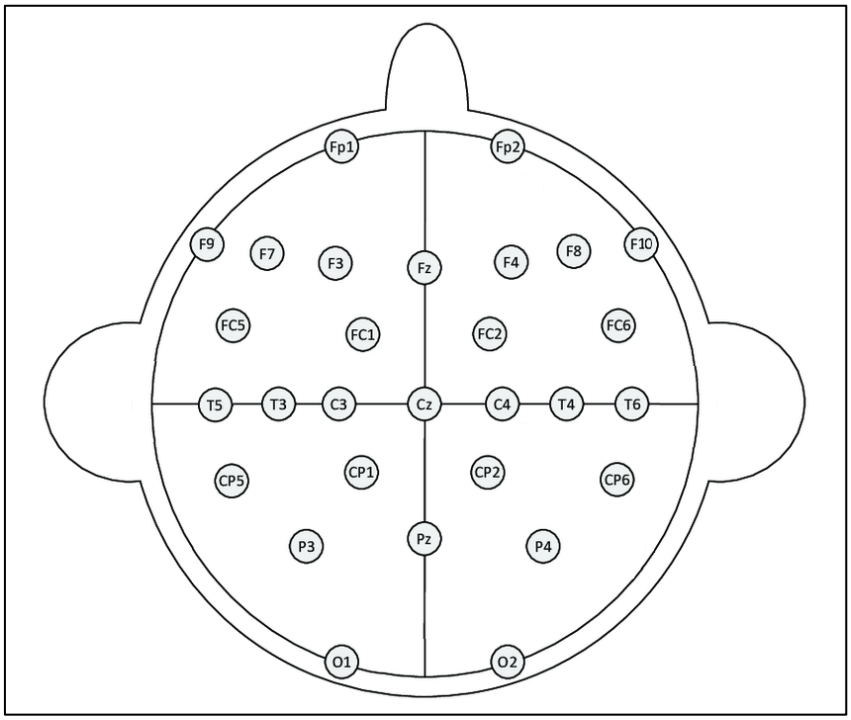

Figure 1 – EEG channels configuration

4. GCN Model for graph classification

The implementation of the above-mentioned classification task using EEG is primarily split into two phases, which are data preprocessing and model classification using a deep learning model.

3.1 Data preprocessing

Once the EEGs were available the first step was to go through the loading and modeling in EEGraph, as mentioned, this is a Python library to model electroencephalograms (EEGs) as graphs [10]. It computes frequency and time-frequency domain connectivity measures. The different available connectivity measures in EEGraph with their default parameters can be observed in Table 1 and its workflow can be observed in Figure 2. It

has applications in the study of neurologic diseases like Parkinson or Epilepsy. The library segments the EEG depending on a chosen window and then converts channels to nodes and edges will represent the chosen connectivity measure between channels. Once the window, the connectivity measure and a threshold are selected, the EEG is modeled. The output is a dictionary of n NetworkX graph objects and the connectivity matrix which is a numpy array of (n x channels x channels) shape.

$$n = \frac{EEG\ Length\ (s)}{Window\ Length(s)}$$

In our case we've set the pre-ictal duration to one hour and the rest of the time as interictal to start with a binary classification (although a second label set was created for ictal stages). We've used a window of 4s, and we've obtained a total of 126937 graphs as it can be observed in Table 2.

Then we've to load the data into a form that can be used by the StellarGraph library [11]. The StellarGraph library offers state-of-the-art algorithms for graph machine learning, making it easy to discover patterns and answer questions about graph-structured data. It can solve our problem of classification of whole graphs.

Connectivity Measure	Threshold	Frequency
Cross Correlation	0.50	No
Pearson Correlation	0.70	No
Squared Coherence	0.65	Yes
Imaginary Coherence	0.40	Yes
Corrected Cross Correlation	0.10	No
Weighted Phase Lag Index (WPLI)	0.45	Yes
Phase Locking Value (PLV)	0.80	Yes
Phase Lag Index (PLI)	0.10	No
Phase Lag Index bands (PLI)	0.10	Yes
Directed Transfer Function (DTF)	0.30	Yes
Power Spectrum	0.25	Yes
Spectral Entropy	0.25	Yes
Shannon Entropy	0.25	No

Table 1 – EEGraph Connectivity measures

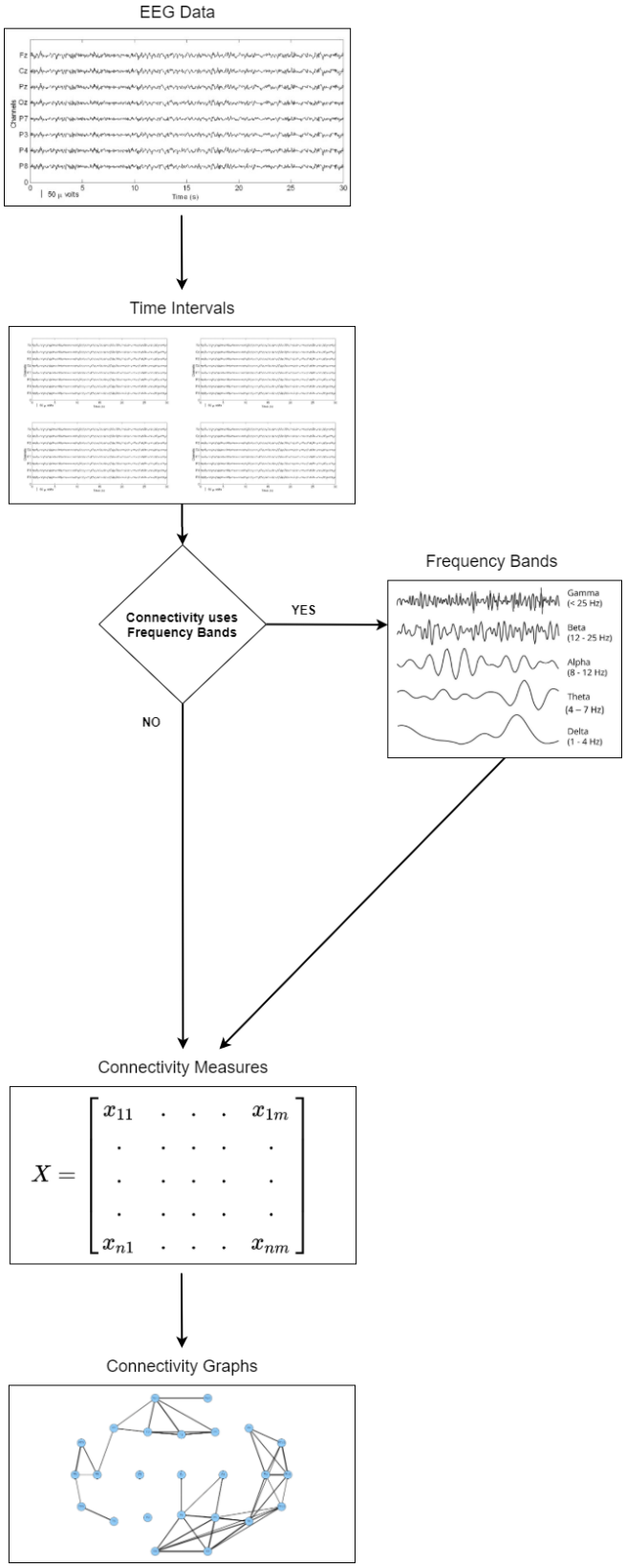

Figure 2 – EEGraph Workflow

Graph-structured data represent entities as nodes (electrodes) and relationships between them as edges (in this case, if the connectivity matrix is above a specific threshold) and can include data associated with them as attributes. StellarGraph supports analysis of many kinds of graphs: in this case we will we working with a homogeneous graph with features (a homogeneous graph is one with only one type of node and one type of edge) where graph features will be the patients age and sex the edges will have weight corresponding to the Pearson correlation, which is the selected connectivity measure, between the nodes.

Patient_id	File	Total_Graphs_Corrected	age_years	gender	rec_time_minutes	rec_time_seconds
PN00	PN00-1.edf	2923	55	Male	198	11880
	PN00-2.edf					
	PN00-3.edf					
	PN00-4.edf					
	PN00-5.edf					
PN01	PN01-1.edf	12140	46	Male	809	48540
PN03	PN03-1.edf	21805	54	Male	752	45120
	PN03-2.edf					
PN05	PN05-2.edf	5433	51	Female	359	21540
	PN05-3.edf					
	PN05-4.edf					
PN06	PN06-1.edf	10867	36	Male	722	43320
	PN06-2.edf					
	PN06-3.edf					
	PN06-4.edf					
	PN06-5.edf					
PN07	PN07-1.edf	7862	20	Female	523	31380
PN09	PN09-1.edf	6165	27	Female	410	24600
	PN09-2.edf					
	PN09-3.edf					
PN10	PN10-1.edf	16857	25	Male	1002	60120
	PN10-2.edf					
	PN10-3.edf					
	PN10-4.5.6.edf					
	PN10-7.8.9.edf					
	PN10-10.edf					
PN11	PN11-1.edf	2170	58	Female	145	8700
PN12	PN12-1.2.edf	5491	71	Male	246	14760
	PN12-3.edf					
	PN12-4.edf					
PN13	PN13-1.edf	7799	34	Female	519	31140
	PN13-2.edf					
	PN13-3.edf					
PN14	PN14-1.edf	18419	49	Male	1408	84480
	PN14-2.edf					
	PN14-3.edf					
	PN14-4.edf					
PN16	PN16-1.edf	4393	41	Female	303	18180
	PN16-2.edf					
PN17	PN17-1.edf	4613	42	Male	308	18480
	PN17-2.edf					
Total		126937				

Table 2 – Data available

Once we've converted our NetworkX graphs coming from EEGraph we add the sex and age as features and convert them into a list of StellarGraphs, where each object has properties like those in Figure 3.

```
StellarGraph: Undirected multigraph
 Nodes: 29, Edges: 200

 Node types:
  default: [29]
    Features: float32 vector, length 3
    Edge types: default-default->default

 Edge types:
    default-default->default: [200]
        Weights: range=[0.701485, 0.98317], mean=0.847474, std=0.0782338
        Features: float32 vector, length 1
```

Figure 3 – StellarGraph Info

After we've our complete set of data converted to a list of StellarGraphs, we can check some summary statistics like those in Table 3 and a histogram of the number of edges in each graph in Figure 4 confirming that the graphs have an average of 92 edges.

	NODES	EDGES
COUNT	126937	126937
MEAN	29	91.9
STD	0	64.3
MIN	29	0
25%	29	52
50%	29	74
75%	29	104
MAX	29	406

Table 3 – Summary statistics for StellarGraphs list

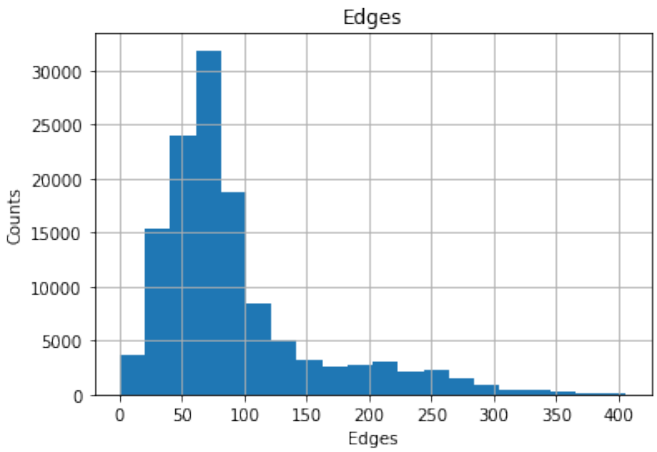

Figure 4 – Histogram StellarGraph Info

After we've checked our preprocessed dataset statistics, we need to split our data into random training, validation and test sets. We are going to use 60% of the data for training, 20% for validation and the remaining 20% for testing.

Given the defined data split into train, validation and test sets, we then create a StellarGraph.PaddedGenerator generator object that prepares the data for training in Keras. A generator just encodes the information required to produce the model inputs. We create data generators suitable for training at Keras model by calling the generator's flow method specifying the train, validation and test data.

3.2 Model preparation and training

We are now ready to create a Keras graph classification model using StellarGraph's GraphClassification class together with standard Keras layers. The input is the list of graph represented by its adjacency and node features matrices comprised in a StellarGraph. The first two layers are Graph Convolutional as in [6] with each layer having 64 units and 'relu' activations and a dropout layer with 0.2 to avoid overfitting. The next layer is a mean pooling layer where the learned node representation is summarized to create a graph representation. The graph representation is input to two fully connected layers with 32 and 16 units respectively and 'relu' activations. The last layer is the output layer with a single unit and sigmoid activation for our binary classification problem, all the model summary and its architecture can be observed in Figure 5 and Figure 6.

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, None, 3)]    0           []

 dropout (Dropout)              (None, None, 3)      0           ['input_1[0][0]']

 input_3 (InputLayer)           [(None, None, None)  0           []
                                ]

 graph_convolution (GraphConvol  (None, None, 64)    256         ['dropout[0][0]',
 ution)                                                           'input_3[0][0]']

 dropout_1 (Dropout)            (None, None, 64)     0           ['graph_convolution[0][0]']

 graph_convolution_1 (GraphConv  (None, None, 64)    4160        ['dropout_1[0][0]',
 olution)                                                         'input_3[0][0]']

 input_2 (InputLayer)           [(None, None)]       0           []

 global_average_pooling1d (Glob  (None, 64)          0           ['graph_convolution_1[0][0]',
 alAveragePooling1D)                                              'input_2[0][0]']

 dense (Dense)                  (None, 32)           2080        ['global_average_pooling1d[0][0]'
                                                                 ]

 dense_1 (Dense)                (None, 16)           528         ['dense[0][0]']

 dense_2 (Dense)                (None, 1)            17          ['dense_1[0][0]']

==================================================================================================
Total params: 7,041
Trainable params: 7,041
```

Figure 5 – Model summary

We can now train the model using the model's fit method. First, we specify some important training parameters such as the number of training epochs (20) and early stopping to avoid overfitting. We'll be using Adam optimizer with a learning rate of 0.0001, binary crossentropy as our loss function and accuracy as our metric always taking into account that we're working with an unbalanced dataset, other metrics were calculated and plotted as well.

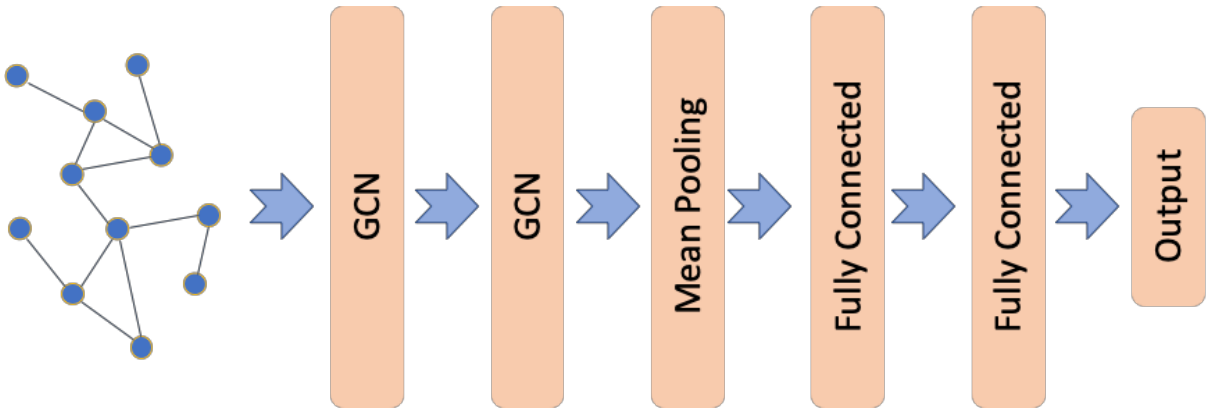

Figure 6 – Model Architecture for graph classification

After compiling our model, we were able to run it, however we realized that accuracy is always around 0.71 which is the proportion of the interictal class which means that the model is not able to learn as we can see in Figure 7 and in the metrics in Figure 8.

Given the fact that this is occurring we've tried to run a different approach based on a Deep Graph CNN architecture was proposed in [12] (Figure 9) using the graph convolutional layers from [6] but with a modified propagation rule. DGCNN introduces a new SortPooling layer to generate a representation (also known as embedding) for each given graph using as input the representations learned for each node via a stack of graph convolutional layers. The output of the SortPooling layer is then used as input to one-dimensional convolutional, max pooling, and dense layers that learn graph-level features suitable for predicting graph labels as we can see in Figure 9 .

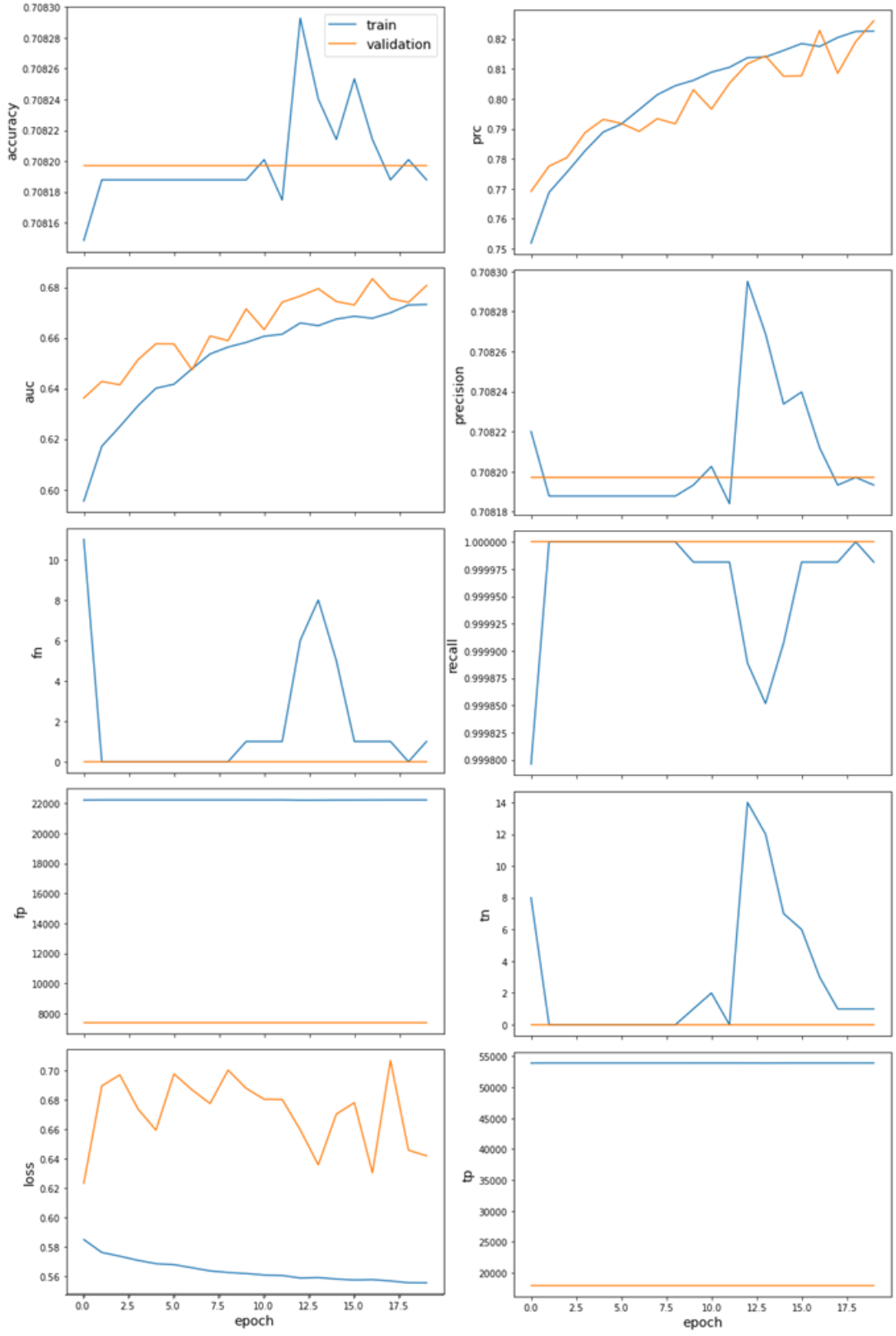

Figure 7 – Learning curves for GCN model

```
Test Set Metrics:
        loss: 0.6437
        tp: 17979.0000
        fp: 7409.0000
        tn: 0.0000
        fn: 0.0000
        accuracy: 0.7082
        precision: 0.7082
        recall: 1.0000
        auc: 0.6744
        prc: 0.8188

Val Set Metrics:
        loss: 0.6419
        tp: 17979.0000
        fp: 7408.0000
        tn: 0.0000
        fn: 0.0000
        accuracy: 0.7082
        precision: 0.7082
        recall: 1.0000
        auc: 0.6807
        prc: 0.8261

Train Set Metrics:
        loss: 0.6421
        tp: 53937.0000
        fp: 22225.0000
        tn: 0.0000
        fn: 0.0000
        accuracy: 0.7082
        precision: 0.7082
        recall: 1.0000
        auc: 0.6781
        prc: 0.8232
```

Figure 8 – GCN model metrics

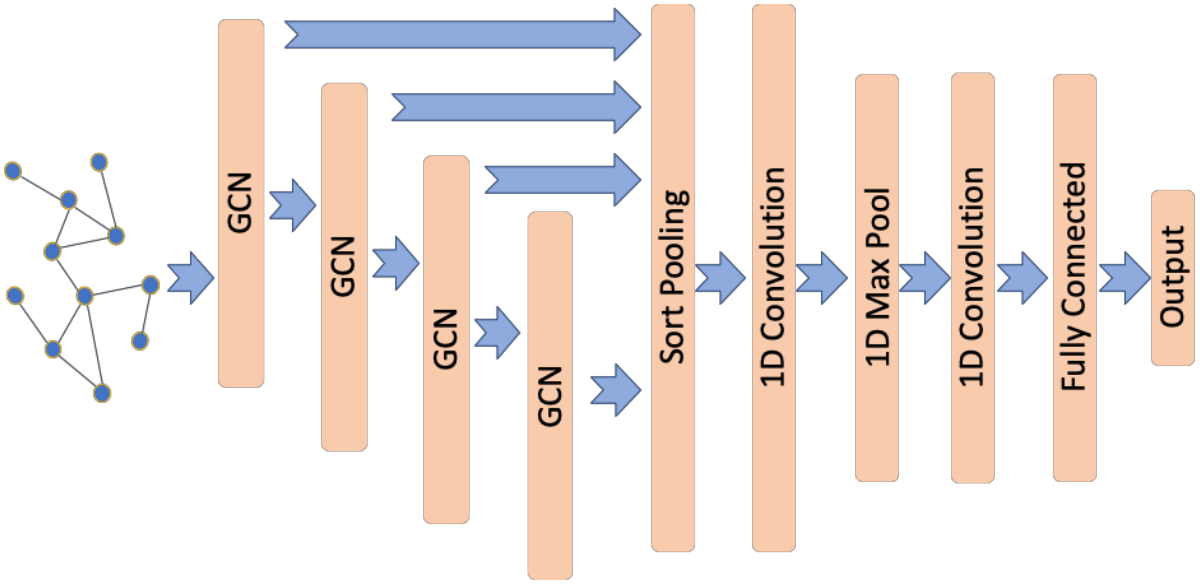

Figure 9 – Model architecture for DGCNN

We can also observe the learning curves in Figure 10 and the metrics in Figure 11 that after 10 epochs the model is not able to learn either.

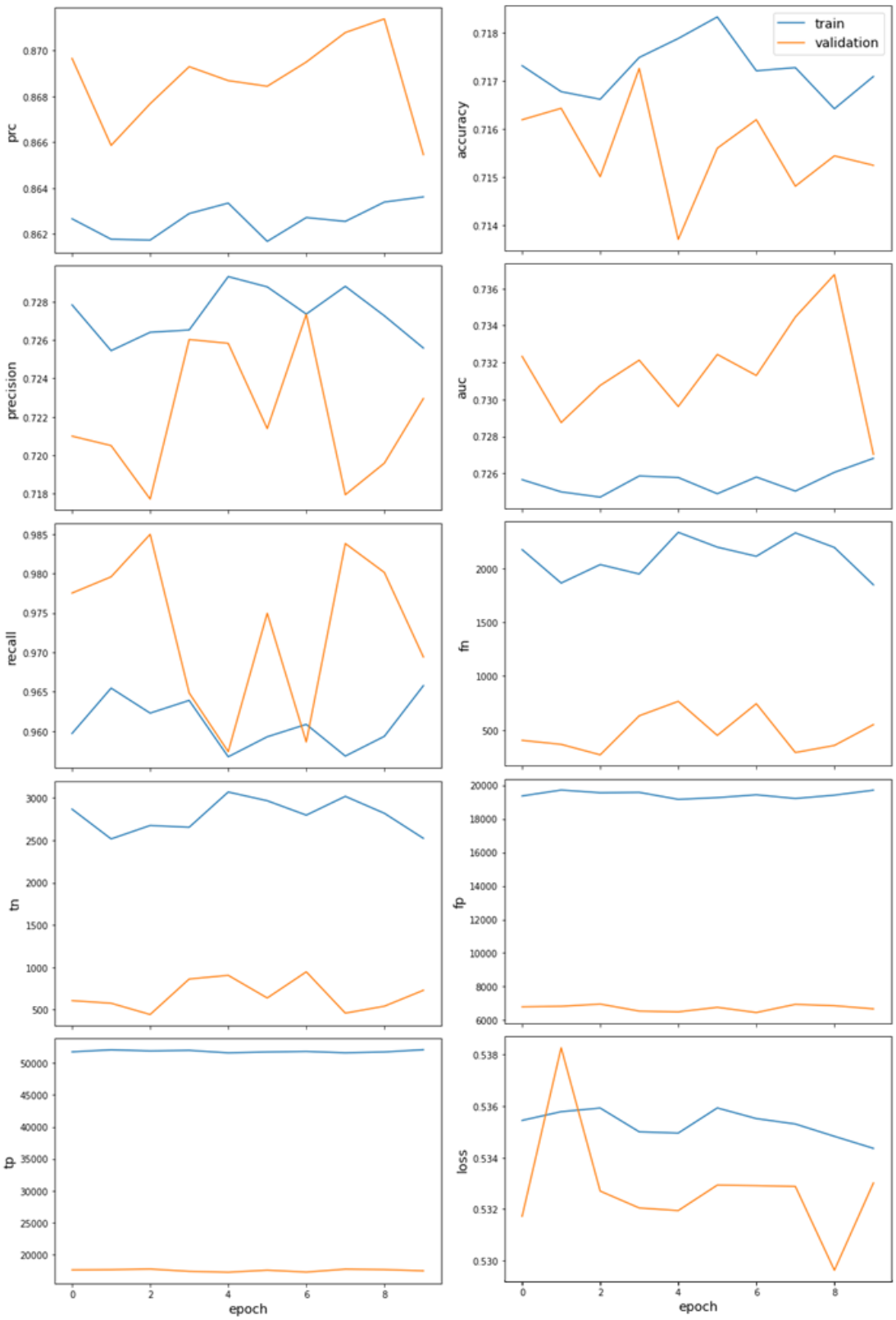

Figure 10 – Learning curves for DGCNN model

```
Test Set Metrics:
        loss: 0.5365
        tp: 17463.0000
        fp: 6635.0000
        tn: 774.0000
        fn: 516.0000
        accuracy: 0.7183
        precision: 0.7247
        recall: 0.9713
        auc: 0.7222
        prc: 0.8611

Val Set Metrics:
        loss: 0.5330
        tp: 17429.0000
        fp: 6679.0000
        tn: 729.0000
        fn: 550.0000
        accuracy: 0.7152
        precision: 0.7230
        recall: 0.9694
        auc: 0.7270
        prc: 0.8655

Train Set Metrics:
        loss: 0.5318
        tp: 52412.0000
        fp: 19896.0000
        tn: 2329.0000
        fn: 1525.0000
        accuracy: 0.7187
        precision: 0.7248
        recall: 0.9717
        auc: 0.7290
        prc: 0.8657
```

Figure 11 – DGCNN model metrics

5. Conclusions

After trying to train both models (GCN and DGCNN) with different archictectures and different ways of generating the StellarGraphs, it was impossible to make it work, we tried processing the data in other ways:

- Excluding the sex and age of the patients
- Using a binary adjacency matrix so that all the weight of the existing edges was 1 and using as features de correlation matrix generated by EEGraph

At the end we were not able train the model, however we've used this same work-flow of preprocessing the data with EEGraph and training with StellarGraph in other EEG dataset at CEIEC from Parkinson disease (PD) patients and the results were a complete success.

In that case we used a dataset of EEG tests over 11 patients with PD and 11 healthy people each with three different tests (resting, left and right finger tapping). EEGs used 64 channels and the 10-20 system. The frequency is 512 Hz that is one measure every 1.9531 milliseconds. From the PD patients, there were two sets of EEGs pre and post Levodopa (also called L-dopa) which is the most prescribed medicine for Parkinson's.

In that case using the binary adjacency matrix so that all the weight of the existing edges was 1 and using as features de correlation matrix we were able to train the model

for the Parkinson-no_Parkinson case and obtain the results in Table 4 after the KerasTuner hyperparameter optimization and a 0.96 ± 0.02 average accuracy after cross validation (accuracy in this case worked because we were working with a balanced dataset.

Metric	Test	Validation	Train
loss	0.095	0.145	0.124
acc	0.980	0.957	0.960

Table 4 – Graphs PD project metrics

Finally, we concluded that this workflow is not suitable for epilepsy stages prediction in this dataset given that fact that we weren't able to train the models even preprocessing the data in different ways, while for PD the workflow worked without any important issued. This lesson learned is also an important outcome which will allow other researchers in the future to avoid this workflow for epilepsy prediction on the Siena Scalp EEG Database through the use of EEGraph and StellarGraph.

References

1. Tsiouris, K. M., Pezoulas, V. C., Zervakis, M., Konitsiotis, S., Koutsouris, D. D., & Fotiadis, D. I. (2018). A long short-term memory deep learning network for the prediction of epileptic seizures using EEG signals. Computers in biology and medicine, 99, 24-37.
2. Chen, X., Zheng, Y., Niu, Y., & Li, C. (2020, July). Epilepsy Classification for Mining Deeper Relationships between EEG Channels based on GCN. In 2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL) (pp. 701-706). IEEE.
3. Zeng, D., Huang, K., Xu, C., Shen, H., & Chen, Z. (2020). Hierarchy graph convolution network and tree classification for epileptic detection on electroencephalography signals. IEEE Transactions on Cognitive and Developmental Systems.
4. Dissanayake, T., Fernando, T., Denman, S., Sridharan, S., & Fookes, C. (2021). Geometric Deep Learning for Subject-Independent Epileptic Seizure Prediction using Scalp EEG Signals. IEEE Journal of Biomedical and Health Informatics.
5. Li, Y., Liu, Y., Guo, Y. Z., Liao, X. F., Hu, B., & Yu, T. (2021). Spatio-Temporal-Spectral Hierarchical Graph Convolutional Network With Semisupervised Active Learning for Patient-Specific Seizure Prediction. IEEE Transactions on Cybernetics.
6. Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.
7. Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems, 29, 3844-3852.
8. Detti, P. (2020). Siena Scalp EEG Database (version 1.0.0). PhysioNet. https://doi.org/10.13026/5d4a-j060.
9. Detti, P., Vatti, G., & Zabalo Manrique de Lara, G. (2020). EEG Synchronization Analysis for Seizure Prediction: A Study on Data of Noninvasive Recordings. Processes, 8(7), 846.
10. Nogales, A., Maitín, A. M, Chazarra, P. (2021). EEGraph Library. GitHub Repository. https://github.com/ufvceiec/EEGRAPH.
11. Data61, C. (2018). StellarGraph Machine Learning Library. GitHub Repository. https://github.com/stellargraph/stellargraph.
12. Zhang, M., Cui, Z., Neumann, M., & Chen, Y. (2018, April). An end-to-end deep learning architecture for graph classification. In Thirty-Second AAAI Conference on Artificial Intelligence.
13. Maitín, A. M., García-Tejedor, A. J., & Muñoz, J. P. R. (2020). Machine Learning Approaches for Detecting Parkinson's Disease from EEG Analysis: A Systematic Review. Applied Sciences, 10(23), 8662.
14. Zhao, Y., Dong, C., Zhang, G., Wang, Y., Chen, X., Jia, W., ... & Zheng, Y. (2021). EEG-Based Seizure detection using linear graph convolution network with focal loss. Computer Methods and Programs in Biomedicine, 208, 106277.
15. Wagh, N., & Varatharajah, Y. (2020, November). Eeg-gcnn: Augmenting electroencephalogram-based neurological disease diagnosis using a domain-guided graph convolutional neural network. In Machine Learning for Health (pp. 367-378). PMLR.