Final Report

GCN-Explain-Net: An Explainable Graph Convolutional Neural Network (GCN) for EEG-based Motor Imagery Classification and Demystification

Qin Hu, Rory O' Keeffe

May 2021

N.B. Our code is publicly available at: `https://github.com/erinqhu/EEG-motor-imagery.git`

Introduction

Biometrics such as Electroencephalography (EEG) signals have drawn substantial interest in decoding brain activities, such as classifying emotions and motor intention. Motor Imagery EEG signals have been extensively researched in support of normal daily living of disabled individuals, through the underlying neurophysiological signal patterns learned by conventional convolutional neural networks. However, often these networks do not obtain optimal prediction performance because they fail to decipher model attention, as well as to discover complete, detailed relationships among EEG signals recorded from different electrodes.

Literature Survey and Mathematical Background

Most of the current models which decode EEG using Deep Learning, particularly CNNs (e.g. EEG-Net), use the Euclidean coordinates of a given EEG electrode in an analogous way to image coordinates for a regular CNN. However, a graphical representation of intra-cortical connectivity (specifically correlation) can be a more informative input for describing the topological relationship (connectivity) between parts of the network [1]. Alternative connectivity metrics to correlation (e.g. coherence) have yet to be explored for this Graph Convolutional Network (GCN) concept. Meanwhile, Layerwise Relevance Propagation (LRP) has been recently applied to EEG classification problems [2], providing visual insight on inner network mechanics, which can explain classification errors and facilitate important modifications, as well as giving us a higher resolution analysis of neural mechanisms.

Spectral Convolution

As outlined by [3], the convolution in the case of GCN is optimally a spectral, rather than spatial convolution. This type of convolutional filter has the advantage of recognizing local features, which are independent of the Euclidean coordinates, and this is usually preferred for GCNs [4].

Node Localization

Hence, the connectivity matrix (adjacency matrix), W_{ij} can give superior information about node organization than Euclidean coordinates. In our case, W_{ij} is computed using Spearman's power correlation between nodes (electrodes). The diagonal degree matrix is simply $D_{ii} = W_{ij}$, and the Laplacian L is then $L = D - W$. The Laplacian is diagonalized by the Fourier basis U, such that $L = UUT$, is a diagonal matrix of eigenvalues, U is termed the graph Fourier transform. In summary, the pairwise node connectivity is related to the Laplacian, and the Laplacian is related to U, which is then used to transform the input to the spectral convolution domain.

Implementing Convolution

The graph convolution (denoted by $*_G$) of signals x and y, is defined in the Fourier domain as:

$$x *_G y = U((U^T x) \odot (U^T y))$$

To ensure the convolution filter localization and maximize its computational efficiency, a polynomial filter:

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k,$$

is constructed, where θ s a vector of polynomial coefficients. An obvious choice here is the Chebyshev family, as these are often used to design signal processing filters and the coefficients can be learnt recursively. In summary, each convolutional layer contains filters, whose coefficients are the learnable parameters. Networks with such spectral convolution, have analogous architectures to conventional convolution networks - (i) RGB values $\rightarrow$ node spectral features, (ii) spatial localization $\rightarrow$ connectivity between nodes, (iii) learnt NN weights $\rightarrow$ Chebyshev polynomial coefficients. Regarding the last point, the backpropagation algorithm computes the (in our case, cross-entropy) loss, with respect to the filter coefficients. Furthermore, all the usual network parameter choices (pooling, activation function, optimization, learning rate, etc.) apply for GCNs, hence the same code structure as CNNs can be used. Pooling, often called graph coarsening in this context, results in the signal features of well-connected nodes being summed, like localized image information is added in a CNN, to reduce dimensionality.

Technical Details

Dataset
Dataset Name: *PhysioNet EEG Motor Movement/Imagery Dataset*
The EEG Motor Movement/Imagery Dataset includes 64-channel EEG signals collected at a sample rate of 160 Hz from 109 healthy subjects who performed eight different tasks in the 14 experimental runs. For execution data, these tasks are (i) left fist, (ii) right fist open and closed, (iii) both fists, (iv) both feet open and closed. The imagined tasks are the same as though executed, such that we have four classes each of **imaginary** and **executed** movements. Subjects were guided to perform tasks by visual feedback.

Data Preprocessing
Each run includes eight T1 trials and seven T2 trials or the other way around, each of which is slightly over four seconds. We use the first four seconds ($160 * 4 = 640$ timestamps) of each trial to ensure each sample is consistent in size (640×64). To balance the data from T1 and T2, we discarded the last T1 or T2 trial in each run. Therefore, each subject performed 84 trials (7 trials * 4 tasks * 3 runs). A weakness in the previous study [1] was that it didn't band-filter or attempt to remove artifacts (e.g. EOG) - rather it simply fed the raw EEG to the GCN. It has been shown that such EEG signal processing techniques can help the prediction accuracy [5], hence we have band-filtered the signals 0.5-80 Hz, and used Automatic Artifact Removal, driven by ICA. As mentioned previously, the PhysioNet Dataset consists of 109 subjects, however for the purpose of efficient research, this work focuses on a smaller subset of subjects (e.g., first 15 subjects) to save time training.

Model Types

It was decided to approach this EEG classification problem in a parallel manner:

1. simply using the 64 channel, time domain signals as the features, inspired by [1] but with some modifications.

2. computing trial-wise features, in the temporal, spectral and connectivity domains.

Both approaches offer their own advantages, and serve as a natural comparison for one another.

Model 1: GCN with Time Domain Signals (TD-GCN)

Input Format

Before training the neural network in Python, we must do some basic pre-processing, which was implemented in MATLAB. Note that for this model (TD-GCN), (i) left and right were considered as separate classes, and (ii) both fists and both feet were consideres as separate classes. The

above statement is true for both imaginary and executed data. Imaginary and executed data were considered separately in 4-class classification tasks.
Initially, the dimensions of the input are $N_{subjects} \times N_{trials} \times N_{classes} \times N_{samples} \times N_{channels} = 15 \times 21 \times 4 \times 640 \times 64$. However, this is reshaped such that the first four dimensions are combined, giving the resultant dimensions $= 15 * 21 * 4 * 640 \times 64 = 806400 \times 64$. So, all the time samples, for all subjects, all trials, all classes, have been combined into one dimension, N. A corresponding labels vector of $N \times 1$ is then constructed. Finally, the data features and labels are shuffled - meaning that we have N random time samples, each with 64 channel values. This can be visualized as N **instantaneous** brain signal heat maps, such as those shown in Figure 1. Note that both imaginary and executed show the most activity on the middle row of electrodes, through Cz (motor area). We also observe that imaginary patterns, especially in this middle row, are scaled down versions of executed patterns. Prior to feeding the input to the model, the **Spearman** Power correlation -

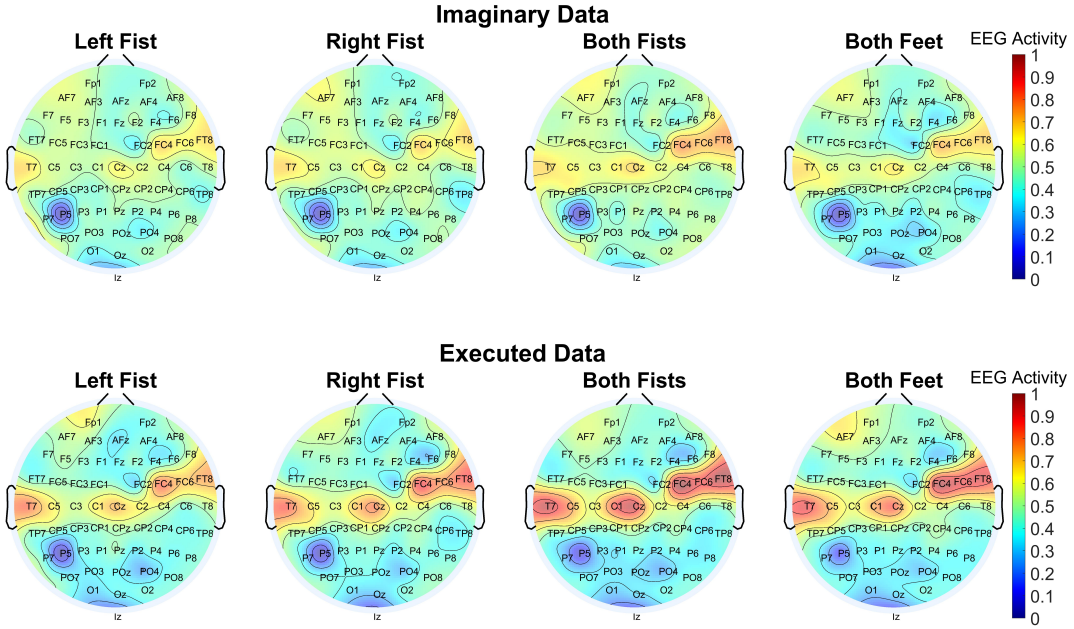


Figure 1: Brain Signal Heat Maps: The color is proportional to the average activation value (RMS) of each electrode, for a given imaginary or executed task. The RMS value was computed across all time samples of a given trial. Then the median RMS (shown) was computed across all subjects' trials.

more appropriate for connectivity than the conventional Pearson Correlation - is computed for all concatenated time samples (not shuffled, in $N \times 64$ format), and the Adjacency and then Laplacian Matrices are then calculated. The Laplacian is then used to perform the graph Fourier transform on the input in the GCN, while the adjacency matrix is used to define how the coarsening operations (akin to pooling in a ConvNet) are applied at each layer. The Adjacency and Laplacian Matrices, derived from Spearman correlation, are shown in Figure 2.
Then, the data is divided into training (80%), validation (10%) and test (10%). The data, labels and other necessary inputs (e.g. Laplacian) for the GCN are saved in CSV files, and will be loaded in the Python code.

Model Details

N.B. This GCN was implemented in TensorFlow, using the GCN with spectral concolution code template, pioneered by [3]. The GCN consists of six graph convolutional layers.
The model architecture is detailed in Figure 3 - the total number of trainable parameters is 355360. The hyperparameters (e.g. dropout rate, regularization coefficient λ) were chosen by trial and error, and a detailed list is given in Figure 4. The network parameters were updated via Stochastic Gradient Descent, using the Adam optimizer. As can be seen in Figure 3, Smooth RELU (Softplus) was used at the activation of each layer and Softmax was used to give the necessary probabilities for the final class prediction. **Cross entropy loss** was selected, with L2 regularization.

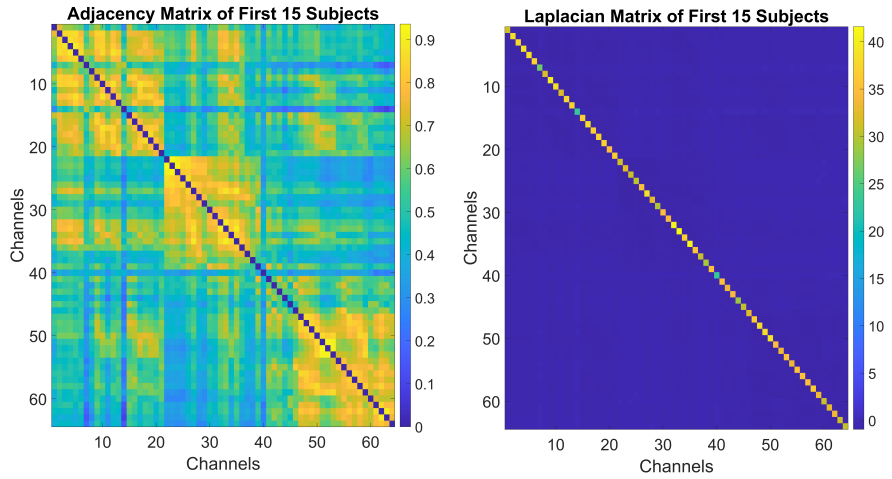


Figure 2: Adjacency (left) and Laplacian Matrices

Layer	Type	Maps	Size	Pooling Size	Activation	Weights	Bias
Softmax	Fully-connected	-	O	-	Softmax	$\frac{N}{64} \times \frac{N}{64} \times F_6 \times O$	O
Flatten	Flatten	-	$\frac{N}{64} \times \frac{N}{64} \times F_6$	-	-	-	-
P6	Max-pooling	F_6	$\frac{N}{32}$	2	-	-	-
C6	Convolution	F_6	$\frac{N}{32}$	-	Softplus	$F_5 \times F_6 \times K$	$\frac{N}{32} \times F_6$
P5	Max-pooling	F_5	$\frac{N}{16}$	2	-	-	-
C5	Convolution	F_5	$\frac{N}{16}$	-	Softplus	$F_4 \times F_5 \times K$	$\frac{N}{16} \times F_5$
P4	Max-pooling	F_4	$\frac{N}{8}$	2	-	-	-
C4	Convolution	F_4	$\frac{N}{8}$	-	Softplus	$F_3 \times F_4 \times K$	$\frac{N}{8} \times F_4$
P3	Max-pooling	F_3	$\frac{N}{4}$	2	-	-	-
C3	Convolution	F_3	$\frac{N}{4}$	-	Softplus	$F_2 \times F_3 \times K$	$\frac{N}{4} \times F_3$
P2	Max-pooling	F_2	$\frac{N}{2}$	2	-	-	-
C2	Convolution	F_2	$\frac{N}{2}$	-	Softplus	$F_1 \times F_2 \times K$	$\frac{N}{2} \times F_2$
P1	Max-pooling	F_1	N	2	-	-	-
C1	Convolution	F_1	N	-	Softplus	$1 \times F_1 \times K$	$N \times F_1$
Input	Input	1	N	-	-	-	-

Figure 3: Model Description, there are 6 graph convolutional layers, with 355360 parameters, trained using stochastic gradient descent.

Batch Size	Regularization	Dropout	Learning Rate	Pooling	Poly Orders
1024	0.001	0.5	0.0001	Max	2

Figure 4: Hyperparameter Choices

Results

After we have finished fitting the model with 50 epochs, the model is finally evaluated using the test data, to give a reliable estimate for accuracy. The validation accuracy and loss curve obtained during training the model with 64 electrodes is shown in Figure 5. As we can see, for the imaginary data, the batch-wise accuracy has saturated at approximately 73% at the final epoch. Further detail about this classification can be obtained by studying the associated confusion matrix, Figure 6. Interestingly, the model accuracy from the executed data is only marginally higher (77%) than from the imaginary data. We also note similarities in the classification mistakes by the model when using imaginary and executed; in both cases, it was (i) more likely to mis-classify left fist as right fist, rather than as both fists, (ii) it was more likely to confuse both fists with both feet, rather than as a single fist. Although these are strong accuracy results, considering this is EEG data, they are not quite high enough to apply Layerwise Relevance Propagation (LRP). LRP had been identified as a possible method of performing interpretable Deep Learning on this EEG dataset, however this technique is usually used when classification accuracy is $> 90\%$ [2].

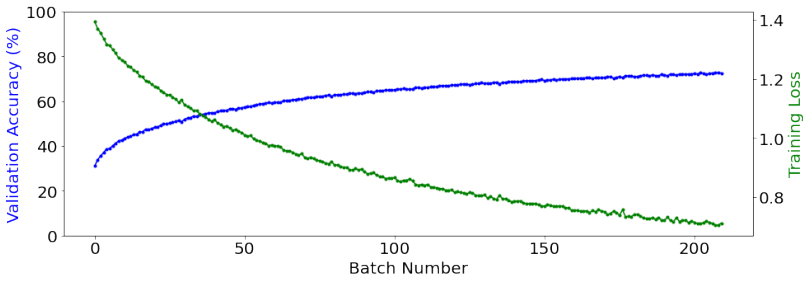


Figure 5: All Electrodes: Validation Accuracy And Training Loss Curves During Training, for Imaginary Data

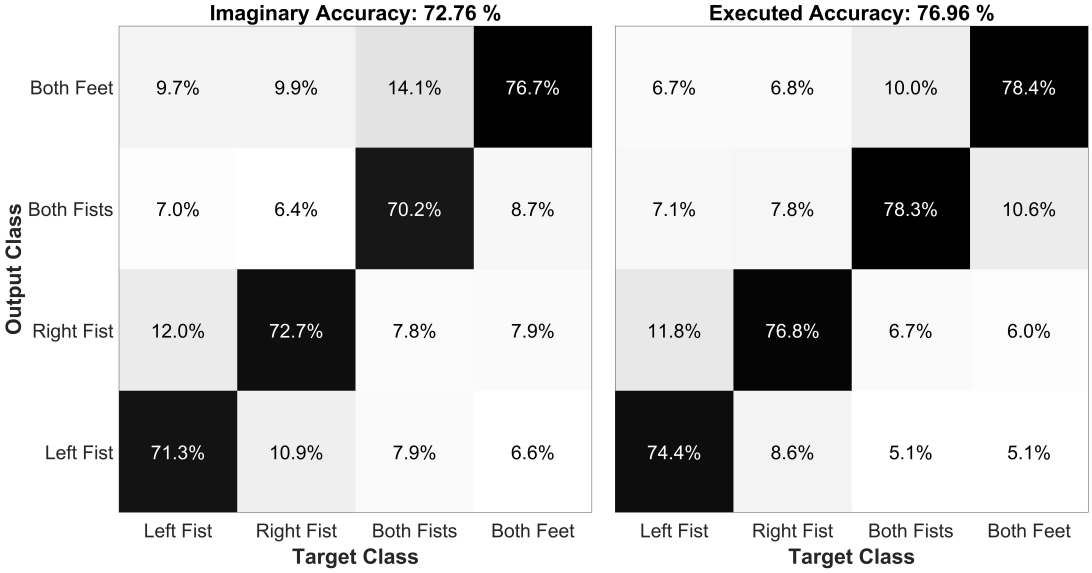


Figure 6: All Electrodes: Confusion Matrix for Imaginary (left) and Executed Data

Therefore, a simpler form of interpretable Machine Learning was employed here - region-wise classification. The 64 electrodes were split into four groups (Figure 7):

1. Cognition (electrodes 1-17)
2. Motor (electrodes 18-36)
3. Command and Integration (electrodes 37-55)
4. Visual (electrodes 56-64)

The network was trained in the very same way for each of the subsets as it had been for the full electrode set, and the classification accuracy was examined. The confusion matrices for each region are shown in Figure 8. We note that the *Visual* electrode set has the lowest classification accuracy, which is unsurprising given that it had the least electrodes in its subset. The accuracies of the other three sets are very comparable with one another. We note that while *Cognition* had higher accuracy, *Motor's* confusion matrix showed more similarity to the overall classification pattern shown in Figure 6, where adjacent classes were also most frequently confused.
It is worth verifying whether the use of such a computationally expensive classifer is really justified. Therefore, a basic SVM classifier was trained and tested on each subject, using just time domain features on all that subject's trials for both imaginary and executed data, Figure 9. It is worth noting that the average classification accuracy over the subjects is far worse than above, for the imaginary (35.1% vs. 73%) and executed data (33.4% vs. 77%).

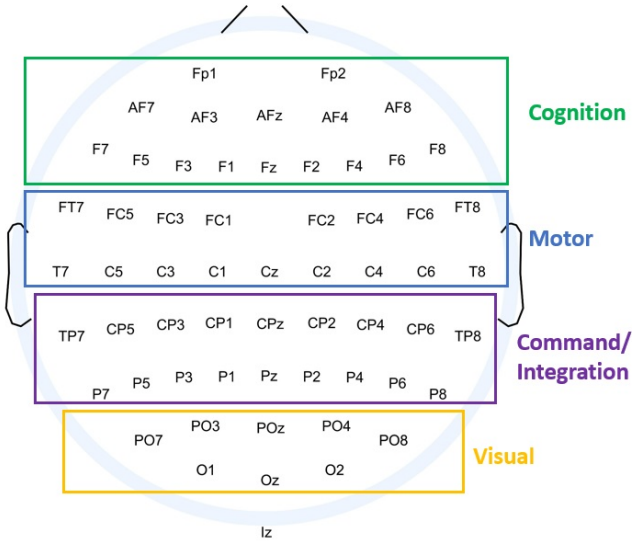


Figure 7: Brain Region Division

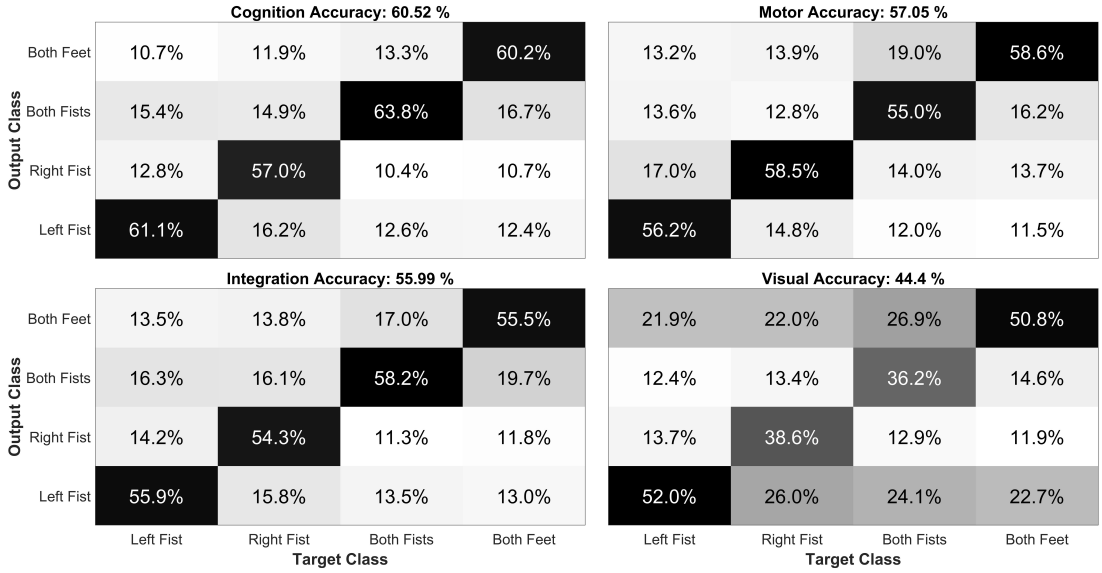


Figure 8: Confusion Matrix Obtained From Training and Testing the GCN on Each Brain Region

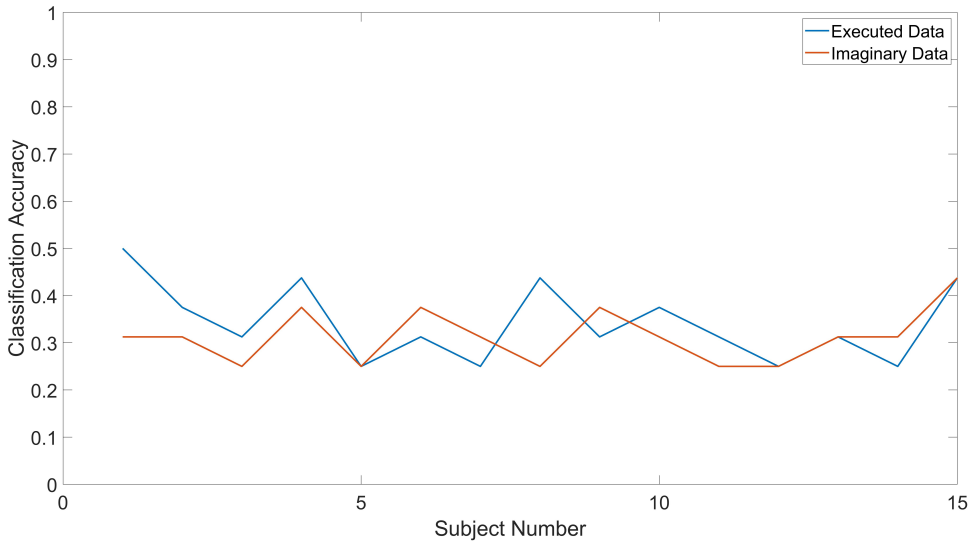


Figure 9: Time Domain Classification Accuracy For Imaginary And Executed Data, when using a basic classifier (SVM)

Model 2: GCN With Diverse Features (F-GCN)

Feature Extraction

In this project, each trial has been considered to be an individual graph, having 64 electrodes as nodes and bidirectional fully connected links between each electrode pair as edges, hence the motor imagery classification has been conducted in a graph classification manner. Before feature extraction, we pre-processed the raw EEG data using StandardScaler normalization based only on the mean and standard deviation of the training data (see train-valid-test segmentation details in the model details subsection under Model 2). To calculate the feature matrix for a trial, we have extracted temporal, spectral, and connectivity features from the normalized EEG signals of the trial for each node(channel). For the time features, we followed [6] to compute Mean Absolute Value (MAV), Variance, Mean Square Root (MSR), Root Mean Square (RMS), Log Detector (LD), Waveform Length (WL), Difference Absolute Standard Deviation Value (DASDV), Zero Crossing (ZC), Skewness, and Kurtosis. For the frequency features, we calculated the mean power density from five conventional frequency bands that consist of Delta (0.5-4 Hz), Theta (4-8 Hz), Alpha (8-12 Hz), Beta (12-35 Hz), and Gamma (>35 Hz). For the connectivity features, we extracted the mean and standard deviation of MS-coherence over each electrode pair (4032 pairs in total excluding self-coherence) from the same five frequency bands. Hence, the dimension of the feature matrix for each trial is 64×25, comprising 10 time features, five frequency features, and 10 connectivity features for each node (channel). The features have been StandardScaler normalized before being fed to the GCN model, since ZC has a significantly large scale compared with other features.

Model Details

Although conventional deep learning techniques such as Long short-term memory (LSTM) and Convolutional Neural Network (CNN) have been substantially implemented in the analogous field of research (EEG-based motor imagery classification), these models ignore the connectivity and strength of the connectivity between electrode pairs. In this project, we propose a Graphical Convolutional Network (GCN) that can not only considers the connectivity and the strength of it, but also exploits the spatial feature extraction power of CNN to decode the underlying neurophysiological features from the motor imagery tasks.

Our chosen GCN model consists of three Chebyshev spectral graph convolutional operators, each of which contains a Chebyshev convolutional (ChebConv) layer, a Batch Normalization layer, and a Rectified Linear Unit (ReLU) layer. The GCN has 32, 64, 128 filters and 3, 4, 5 filter sizes on each ChebConv layer, respectively. In the end, we forward pass the concatenated outputs from global average pooling and global maximum pooling to a dense layer that has 256 cells followed by a Log Softmax activation to obtain the predicting probability of each class (task). The proposed GCN model is compact, in that it **only has 53252 trainable parameters**, compared to TD-GCN (355360) and conventional deep learning models (which can sometimes have >1 million trainable parameters).

For the GCN model implementation, we used PyTorch Geometrics. Under this library, a GCN takes but not limited to three inputs: a feature matrix, an adjacency matrix in coordinate format (COO), and an edge attribute matrix that stores edge weights from each trial. The feature matrix derivation has been stated in detail in the feature extraction subsection under Model 2. The adjacency matrix of each trial is a fully connected matrix excluding self-connections (4032 edges in total). To compute the trial-wise edge attribute, we employed the Spearman correlation coefficient to conduct pairwise comparison between electrodes (nodes).

Results

For model training, we first conduct the train-valid-test split on the current dataset for the normalization before feature extraction and connectivity calculation. Note that for this model (F-GCN), in contrast to TD-GCN:

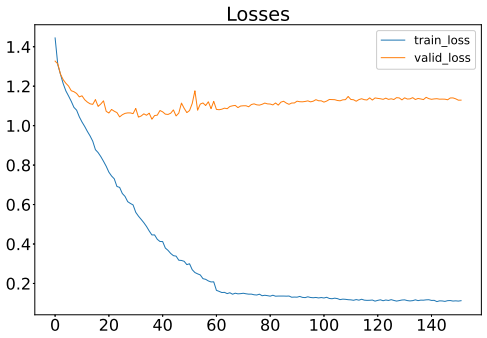
- imaginary and executed were considered together in the classification
- left or right fist were considered as one class for executed (task 1) and another for imaginary (task 2)

- both fists or both feet were considered as one class for executed (task 3) and for imaginary (task 4)

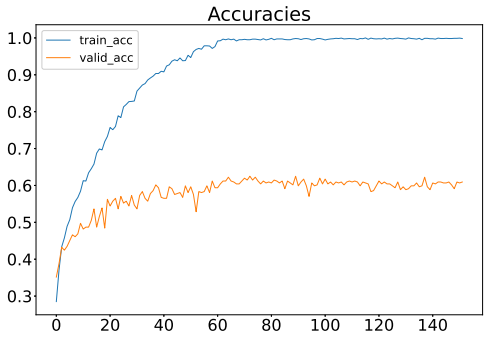
As we have seven imagery and execution trials, we randomly selected five trials ([1, 2, 3, 5, 6]) as training, and respectively consider trial 4 and 6 as validation and test trials. As a result, we derived a train-valid-test split of 72-14-14. The mean and standard deviation of the training set were used to normalize all the datasets, as the raw signals have relatively small amplitudes which may affect feature extraction, electrode connectivity calculation, as well as the training process.

Model training has a mini-batch size of 32, and a learning rate of 4e-4 on Adam optimizer. We exploit learning rate scheduler which will reduce the learning rate by 0.1 after 60 training iterations. Additionally, weight decay regularization and early stopping technique have been utilized to prevent overfitting issues. The patience is set to 80, meaning that the previously saved best model will be taken if the model accuracy does not improve for 80 consecutive iterations. Cross-entropy is considered to be the loss function.

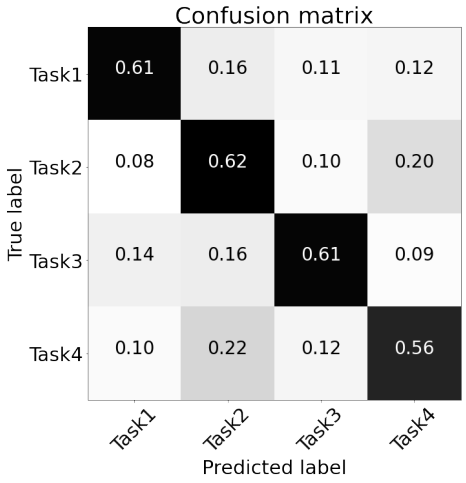
The proposed F-GCN model with diverse features achieves a validation **accuracy of 62.50% (test accuracy: 62.50%, a precision of 60.37%, a recall of 60.00% and a F1-score of 60.08%.** The training process is efficient that the **training per iteration** is **≈1.3s** due to the small number of trainable parameters. The proposed model converges at epoch 90 (**convergence time: ≈117s**). The training and performance details are shown in the loss/accuracy plot (Fig. 10a and Fig. 10b) and confusion matrix (Fig. 10c). From the confusion matrix, we can observe that the F-GCN model is mostly confused between two executed tasks (Task 1 and 3) or two imagery tasks (Task 2 and 4), rather than confusing an imagery task for an execution task.



(a) Losses



(b) Accuracy



(c) Confusion Matrix

Figure 10: GCN with Diverse Features Model Performance: The task of opening/closing one hand is denoted as Task 1; the task of imagining opening/closing one hand is denoted as Task 2; the task of opening/closing both hands is denoted as Task 3; the task of imagining opening/closing both hands is denoted as Task 4.

In this project, we also evaluated the contribution and significance of different types of features (temporal, spectral, or connectivity features) to the overall performance. As can be observed from Table 1, the 10 temporal features contribute the most to the overall performance (51.04%/62.5%=81.66%) followed by the connectivity features (48.18%/62.5%=77.09%). Frequency features have the least contribution of 72.08% (45.05%/62.5%), since Chebyshev convolution is a spectral convolution which is highly capable of extracting frequency features from input data, hence frequency features in the input data are trivial during the training process.

	Accuracy	**Precision**	**Recall**	**F1-Score**
All Features	62.50%	60.37%	60.00%	60.08%
Time Features	51.04%	50.39%	50.28%	50.21%
Frequency Features	45.05%	43.06%	43.06%	42.87%
Connectivity Features	48.18%	49.87%	48.89%	49.00%

Table 1: GCN Performance on Different Features

Comparative Study

The goal of the comparative study is to evaluate the performance and efficiency of GCN, relative to other models. We compare our proposed GCN models (with time-domain signals or with various features) with three commonly-used conventional models; LSTM and CNN are deep learning models, and SVM is conventional classifier. The four-layer LSTM model has 128 cells on each layer followed by a dense layer with 128 cells and a dropout layer with a dropout rate of 0.2. The two-layer CNN model consists of two convolutional modules, each of which has a convolutional layer, a Batch Normalization layer, and a Parametric Rectified Linear Unit (PReLU) layer. The CNN model has 8, 16 filters and 15×9, 15×9 filter sizes on the first and second CNN layers, respectively. There is a Max Pooling layer with 2×2 window size in between CNN layers. The CNN model is ended by a dense layer with 16 cells, followed by a Sigmoid activation and a dropout layer with a 0.2 rate. The LSTM and CNN models are structurally comparable (≈550k trainable gnals. For the SVM, we utilize the same 25 features in the F-GCN model. The results are shown in Table 2.

TD-GCN achieved the best results of all models, followed by the F-GCN model. This is as expected because TD-GCN has more trainable parameters than F-GCN. Recall also the mechanism of sample creation for TD-GCN, where every time point of every trial (of 640 time points) was an instantaneous sample with an associated label. On the other hand, F-GCN extracted features from each trial's duration. This means that TD-GCN had 640 times the number of samples, on which to train and test the model. As the TD-GCN neural network was given more data to learn, it learnt more efficiently. However, although TD-GCN has higher accuracy, F-GCN has some advantages - it has $< \frac{1}{6}$ the number of trainable parameters, its mechanism of pre-computing feautres is arguably more efficient, etc.

It can be observed that F-GCN outperforms conventional models due to its capability of decoding the underlying neurophysiological signal/feature patterns, by considering each trial as a 64-node graph. Interpreting each trial from a graphical angle allows GCN models to learn the connectivity and the strength of connectivity between electrode pairs. It can be seen that SVM achieves decent performance compared to other conventional models, meaning that the extracted features are determinative that capture the patterns from the normalized EEG signals.

	Accuracy	**Precision**	**Recall**	**F1-Score**
***Time Domain GCN (TD-GCN)**	72.75%	72.91%	72.76%	72.75%
GCN with Features (F-GCN)	62.50%	60.37%	60.00%	60.08%
four-layer LSTM	36.11%	36.10%	36.11%	35.82%
two-layer CNN	41.67%	41.79%	41.67%	41.48%
SVM	54.72%	55.12%	54.72%	54.68%

Table 2: Performance of Proposed GCN Models Compared with Conventional Models.

Note: *computed for imaginary data only.

Conclusion

A key aim of this project was to gain insights about neurophysiological mechanisms, through EEG classification using a GCN. Using confusion matrices in a variety of ways, allowed us to draw conclusions about neural mechanisms, during motor imagery and execution:

i The classification patterns for distinguishing tasks from executed data are similar to that for imaginary data, Figure 6. The signal features for imaginary and executed data are similar therefore - this is supported by Figure 1 also.

ii The signal features associated with single limb movements are distinctive from those associated with double limb movements (Figure 6).

iii Given that all region subsets had an accuracy significantly greater than chance (Figure 8), information relevant to a motor task is present in all parts of the brain.

iv Each area's feature set contributes to the overall classification accuracy (Figure 8). Rather than one specific area of the brain containing the discriminative information regarding different motor tasks, different components of the information are spread around the brain.

v The most discriminative components lie in the prefrontal cortex (responsible for Cognition, Figure 8), rather than in the frontal cortex (traditionally responsible for motor planning).

vi The signal features associated with imaginary data are quite distinctive from executed data (Figure 10(c)).

Generally, we can conclude that GCN is well-suited to this EEG classification task, as it has better performance than other neural networks (CNN, LSTM) with a comparable number of trainable parameters. Another important deduction from our results relates to how the GCN **naturally selects** features. TD-GCN had the best accuracy, with simple time domain features being fed to a high-powered GCN. On the other hand, when the features were manually selected beforehand, and fed to a more compact GCN, a worse (but still respectable) accuracy was obtained. This indicates that the TD-GCN is naturally selecting features in a superior way to manual feature selection. Moreover, when given manually selected features, GCN operates relatively well; F-GCN is superior to SVM in this context, in contrast to other Deep Neural Network types. As feature selection has been an open topic and is highly depending on task definition, training an auto-encoder to extract features that optimally represent the EEG signals of each trial can be one of our future works.

References

[1] X. Lun, S. Jia, Y. Hou, Y. Shi, Y. Li, H. Yang, S. Zhang, and J. Lv, "Gcns-net: A graph convolutional neural network approach for decoding time-resolved eeg motor imagery signals," *arXiv preprint arXiv:2006.08924*, 2020.

[2] I. Sturm, S. Lapuschkin, W. Samek, and K.-R. Müller, "Interpretable deep neural networks for single-trial eeg classification," *Journal of neuroscience methods*, vol. 274, pp. 141–145, 2016.

[3] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *arXiv preprint arXiv:1606.09375*, 2016.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[5] J. A. Urigüen and B. Garcia-Zapirain, "Eeg artifact removal—state-of-the-art and guidelines," *Journal of neural engineering*, vol. 12, no. 3, p. 031001, 2015.

[6] Q. Li, P. Dong, and J. Zheng, "Enhancing the security of pattern unlock with surface emg-based biometrics," *Applied Sciences*, vol. 10, no. 2, p. 541, 2020.