Multi-class Motor Imagery Identification using Graph Neural Network

Yagya Modi, Kartik Rai
Department of Computer Science and Engineering, Indian Institute of Technology, BHU, Varanasi

***Abstract*—Brain Component Interface acquires brain signals, analyzes them, and translates them into commands that are relayed to an output device to carry out a desired action. It is one of the most important field of study in today's environment. Not only it helps the people with disabled body function but is also helps the human brain to connect with human made components. Electroencephalography (EEG) is an efficient modality which helps to acquire brain signals corresponds to various states from the scalp surface area. This helps in understanding various brain states and convert them into machine understandable format. Our project highlights on the multi-class identification of EEG-signals via the Graph Neural Networks(GNNs). We will convert our EEG signals into graph objects and compute their similarity via multilayered neural networks for final identification of input data.**

I. Introduction

Brain computer Interface (BCI) has become an essential sector in automated systems, computer-supported physical systems and many more. BCI is the way of generating communication from human brain to the computer and handling complex physical applications. The medical care system has been engulfed with the improvement in computer supported systems, like, prosthetic limbs, mind-controlled home robotization etc., for physically impaired individuals. Motor Imagery is the subdomain in BCI which manages the simulation of motor activities in the brain without actually doing any activity in the real. EEG (Electroencephalogram) signal is the most pertinent, non-obstrusive, multi-channel and fastest modality for recording brain signals.

Methods that are applied to get features for finding neural activities corresponding to motor imagery are like time-frequency domain features and Wavelet transform method(WT). Several issues that we saw in some published papers regarding it are they use very few observations as input for classifiers or they haven't cross validated their proposed classification procedures.

Basic procedure followed in published papers involves: preprocessing, feature extraction, and classification. In feature extraction methods, some approaches that we have seen in papers have certain demerits like Fourier transform preserves the spatial resolution of the signal but loses temporal resolution, the Common Spatial pattern(CSP) relies on the assumption that every class follows the Gaussian distribution, method autoregressive is sensitive to noise content, etc. On reviewing past papers, we saw several other methods relies on basic machine learning techniques like KNN, support vector machine (SVM) [1]

Therefore, the present study that we have decided to use in our project will utilize the applications of Graph Neural Networks for identification of various EEG signals. Various problems can be built around graphs and then can be transformed into suitable graph object to be classified into various classes. Some examples are, text classification, a broadly famous paper known as [2], in which the scientists attempt to use GNN to become familiar with the fine grained word representations dependent on their local structures [3].

II. Graph Similarity using Sim:GNN

We take our inspiration from SimGNN: A Neural Network Approach to Fast Graph Similarity Computation [4] to identify the EEG signals. First we construct our graph and it's features manually, then apply the SimGNN approach on the graphs constructed.

Because of the design of neural network computation, SimGNN has a significant advantage in terms of performance, in terms of effectiveness. However, the neural network architecture must be carefully designed to meet the three properties mentioned below:

1) **Representation-invariant:** By permuting the nodes order, the same graph can be addressed by various distinct adjacency matrices. Such adjustments should have no impact on the calculated similarity score.
2) **Inductive:** The similarity computation should be generalizable to graphs that haven't been seen before, i.e. calculate the similarity score for graphs that aren't part of the training graph pairs.
3) **Learnable:** By changing its parameters through training, the model should be adaptable to any similarity metric.

The approach plans to use two strategy to compute the similarity score:

1) Use attention mechanism to obtain the graph-level embedding by selecting the most relevant parts of the graph and storing the similarity between two graphs.
2) Apply a unique pair wise node comparison method to boost the graph level embedding for stronger interaction in the graph similarity score.

The method is very well versed for similarity computation of pairwise graph objects thus we proceed with pairwise graph matching and identification of the respective graph. For each patient data we compare the model with test data and thus find the accuracy and loss. For identification of a class we take input labeled-data and correctly identify the class it belongs

to, by comparing with each model, the least error producing model is our respective class.

III. Preliminaries

The procedures of project to identify multi-class motor imagery signals are we first pre-processed the data to increment signal-to-noise ratio, removed artifacts and got the signal in beta frequency range.

A. DATASET

This informational collection comprises of EEG information from 9 subjects.The cue-based BCI perspective consisted of four distinctive motor imagery tasks, namely the imagination of movement of the left hand (class 1), right hand (class 2), both feet (class 3), and tongue (class 4). Two meetings on various days were recorded for each subject. Each meeting yielding a total of 288 trials per meeting.
In project we use BCI competition-IV (contains a 4-class MI EEG signal) dataset. It is a 250 Hz and bandpass-filtered between 0.5 Hz and 100 Hz. The information is recorded using 25 channels (22 EEG channels and 3 EOG channels). **Data Recording** : Twenty-two Ag/AgCl electrodes (with inter-electrode distances of 3.5 cm) were utilized to record the EEG; the montage. All signals were recorded mono polarly with the left mastoid serving itself as a reference and the right mastoid as ground. The sensitivity of the amplifier was set to 100 µV. An additional 50 Hz notch filter was applied to suppress line noise.

B. DATA PREPROCESSING

The 3 EOG (Electrooculography) channels present act as noise to our dataset so they are removed. Now the dataset has 22 EEG channels.

1) **Savitzky–Golay Filter**
   Filter is applied to noisy signals to smooth the data, i.e.,to extend the accuracy of the data without distorting the signal tendency. Ordinarily, this digital filter used the method of linear least squares for smoothing the data, which helped with gaining a high signal-to-noise proportion and holded on to the native state of the EEG signal.
   Golay filter needs 3 sources of parameters: polynomial order (k), the noisy signal (x), and its frame size (f).

2) **Butterworth Filter**
   The Butterworth filter gave a frequency response as much flat as possible in the pass band. It is called as "maximally flat" (no ripples) also. On incrementing the filter order, the count of cascaded levels with the filter also increments. In practice, ideal Butterworth's frequency response can't be obtained since excessive ripples are produced in the passband [5].

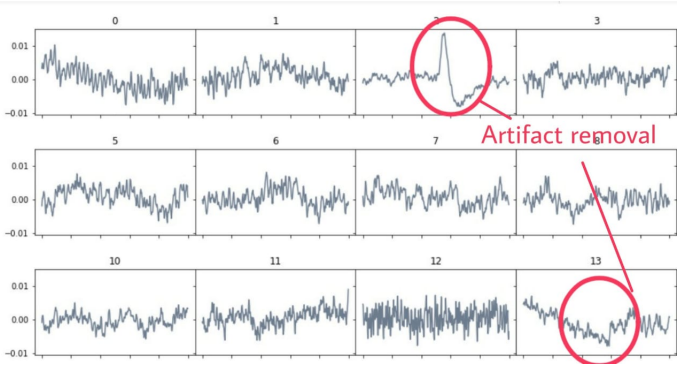


Fig. 1. ICA Artifact Removal

3) **Artifacts Removal (ICA)**

   Recorded signal data from scalp is contaminated with the noise and artifacts. So they affect the overall classification quality of the signal. ICA-based artifact removal segregate and eliminate a wide assortment of artifacts from recorded EEG signal by using linear decomposition.
   The technique uses spatial filters inferred by the ICA algorithm, and does not need a reference channel for every artifact source. When the autonomous time courses of various brain and artifact sources are drawn out from the EEG signal, artifact-corrected EEG sequence can be inferred by taking out the shares of the artifactual sources.
4) **Band Pass Filter** to get the frequency of beta band (12 to 30 Hz).
5) **Segmentation** - Segmented each motor imagery file of every subject into a chunk of 1000 rows * 22 channels, for the next stages.

C. HAUSDORFF DISTANCE

Hausdorff distance is the maximum distance of a set to the nearest point in the other set . Let X and Y be two non-empty subsets of a metric space (M, d). We define their Hausdorff distance by

$$[h]d_h(X,Y) = \max\{supd_{(xX)}(x,Y), supd_{(y\epsilon Y)}(X,y)\} \quad (1)$$

Here our metric space is a 2D tensor and the values are adjusted accordingly. Its is factored to signify a observational changes.

The feature matrix is taken as set of points for each graph and we compute the Hausdorff distance between two graphs by computing Hausdorff distance between their node labelling.

IV. The Proposed Approach

A. GRAPH DESIGNING AND NODE LABELLING

The procedures of project to classify multi-class motor imagery signals are we first pre-processed the data to increment signal-to-noise ratio, removed artifacts and got the

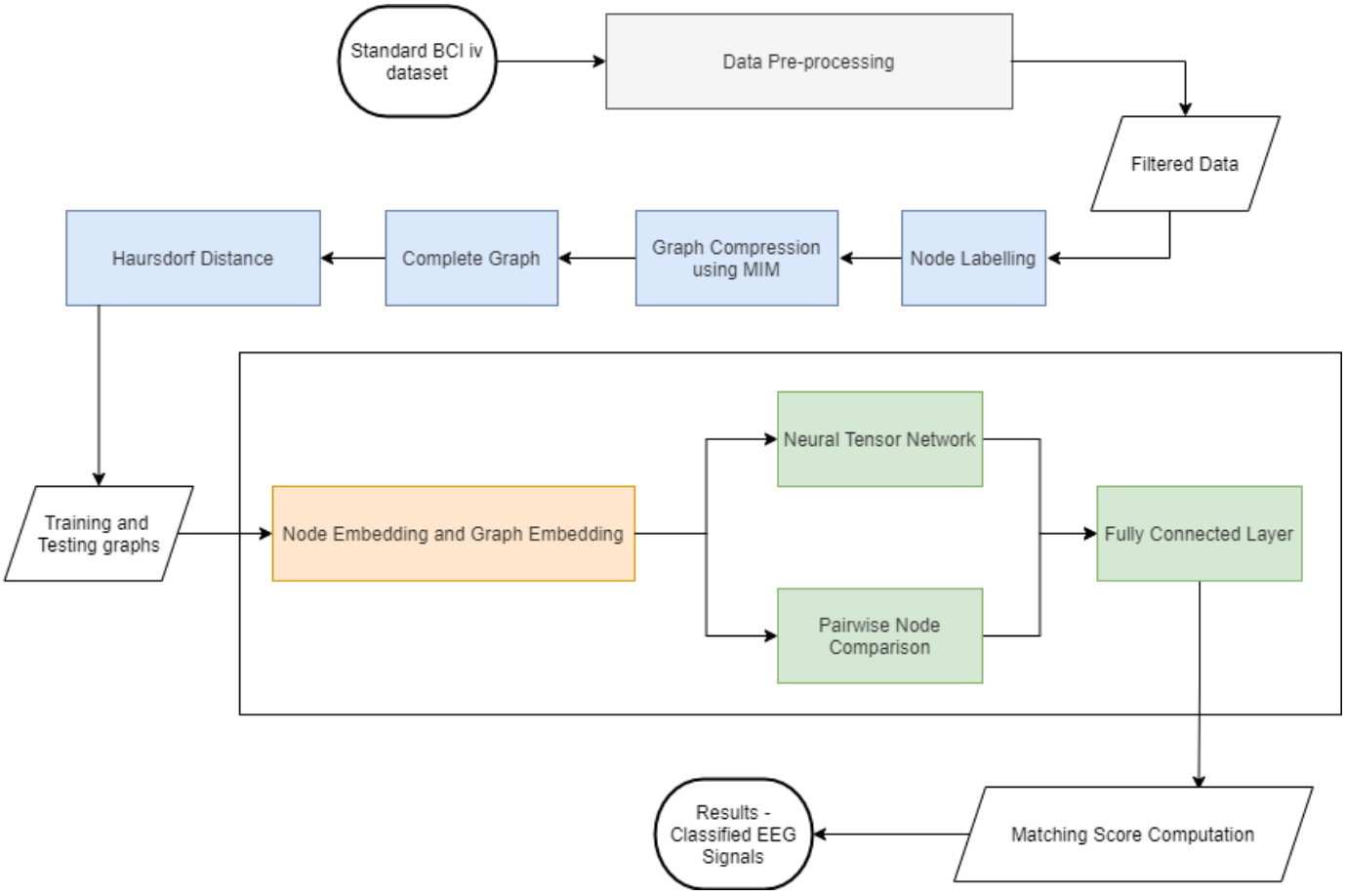


Fig. 2. Block Diagram for proposed method for Multi-class Motor Imagery Identification using Graph Neural Network

signal in beta frequency range. We extracted three types of features for every channels of the filtered EEG signal - Fractal features, Entropy features, and Higher order statistical features. Considering every channel as a node, we designed the complete graph with weighted edges, whose novel approach is discussed in paper. To reduce the computational complexity we selected specific feature containing channels [6]. It is a automatic channel selection technique used for classification.

B. NODE AND GRAPH EMBEDDING

We take our node labels as our feature vectors and we feed them into the GCN [] layer to for node embeddings as mentioned before. These node embeddings represent fine grain interaction between neighboring nodes, the node embedding via self attention mechanism are feed-ed into another GCN layer to form graph embedding. These graph embedding inherits the node-node and node-graph interaction between the two observation graphs.

C. GRAPH-GRAPH INTERACTION VIA NEURAL TENSOR NETWORK

The graph embedding is thus fed into the Neural Tensor Layer to find a similarity score matrix. It facilitates mediated interaction between two entity vectors via a tensor. With a given knowledge base NTN predicts unseen relationships between the graph embeddings thus giving fine graph-graph interaction score.

D. IDENTIFICATION

The similarity score matrix is passed through a Fully-Connected Neural Network layer to find a single similarity score. The final layer is a sigmoid layer which gives a similarity score. For identification, we take test data of a random class for each subject, then we find the score with each class model, and obtain a confusion matrix. We compute a confusion matrix to find the accuracy and thus evaluate our performance.

Details of all headlines in proposed approach is explained below:

V. NODE LABELLING

There is 22 channels data corresponding to the 22 electrodes attached to brain different anatomical regions, in every segmented EEG data file. We treat every channel as a node. Corresponding to every node, we extract 17 features of the node. There are three types of features that we extract -

1) Various order Statistical features

2) Fractal features
3) Entropy features

All the extracted features and their motives is discussed below:

A. STATISTICAL FEATURES

We have extracted 8 statistical features.

Kurtosis

Kurtosis is used to depict distribution. Kurtosis estimates extreme values in on or the other tail. Larger kurtosis of a distribution shows data of tail that are commonly more extreme than the tails of the ordinary distribution.

$$K = n\frac{\sum_{i=1}^{n}(X_i - X_{avg})^4}{(\sum_{i=1}^{n}(X_i - X_{avg})^2)^2} - 3$$

Hurst Exponent

Hurst Exponent is measure of long-term memory of the time series data. If hurst exponent(G) is in range 0.5-1, then time-seies data is long term positive correlation. Else if the G values is in range 0-0.5, then data type is of long-range switching in neighbouring pairs between low and high values.

Detrended Fluctuation Analysis (DFA)

Detrended Fluctuation Analysis is a technique of finding statistical self-affinity. DFA can also be applied to signals whose hidden statistics (like variance, mean) or dynamics are non-stationary.

$$X_t = \sum_{i=1}^{t}(x_i - x_{mean})$$

Where X_t is called cumulative sum. Let Y_t show the resultant piece-wise sequence of straight-line fits. The fluctuation, is computed as:

$$F(n) = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(X_t - Y_t)^2}$$

Other used statistical features are Bi-spectrum, Hjorth mobility and complexity,Fisher information, Root mean square value and total variation

B. FRACTAL FEATURES

Fractal measurement is a proportion giving a statistical measure of complexity by looking at how detail in a fractal pattern changes with the scale at which it is estimated. It has likewise been portrayed to the space-filling capacity of a fractal pattern which gives us a measure of how a fractal scales with respect to the space it is embedded in.

Higuchi Fractal dimension

Higuchi's method is iterative in nature, similar to the box counting. It is used more often when we want to use time-series data as objects. Rebuilt time sequence is:

$$x_m^k = x(m), x(m+k), x(m+2k), ..., x(m + \left[\frac{N-m}{k}\right]k)$$

where k is the span between two neighbouring time signal series.s For each x_m^k , the mean length is:

$$L_m(k) = \frac{\sum_{i=1}^{\frac{N-m}{k}} |x(m+ik) - x(m+(i-1)k|\,(N-1)}{\left[\frac{N-m}{k}\right]k}$$

Subsequently, for the D being the Higuchi fractal dimension. The overall mean length of the discrete time series signal data is shown as follows [7]:

$$lnL(k) \propto Dln\frac{1}{k}$$

Katz Fractal Dimension

Katz dimension of a node is a measure of centrality in a network. It measures the relative degree of influence of a node within a social network.

For a signal (x_i, y_i) of length N, the Katz fractal dimension is shown as follows [7]:

$$D = \frac{log(N)}{log(N) + log(\frac{d}{L})}$$

where L is the length of the waveform, and d is the maximum distance between the initial point (x_1, y_1) to the other points. They can be defined as follows:

$$L = \sum_{i=0}^{N-2}\sqrt{(y_{i+1} - y_i)^2 + (x_{i+1} - x_i)^2},$$

$$d = max(\sqrt{(x_i - x_1)^2 - (y_i - y_1)^2}).$$

Sevcik Fractal Dimension

It measures the complexity and randomness of the waveform. Supposing the signal consists of a series of points (x_i, y_i), the length of signal is N. Normalize the signal first:

$$x_i^* = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

$$y_i^* = \frac{y_i - y_{min}}{y_{max} - y_{min}}$$

Then, the Sevcik fractal dimension D can be calculated as follows [7]:

$$D = 1 + \frac{ln(L) + ln(2)}{ln\,[2(N-1)]}$$

where L is the length of waveform, shown as follows:

$$L = \sum_{i=0}^{N-2}\sqrt{(y_{i+1}^* - y_i^*)^2 + (x_{i+1}^* - x_i^*)^2}$$

Petrosian Fractal Dimension

For the time series data is $x_1, x_2, ..., x_N$. Let the waveform signal contain a series of points $y_1, y_2, ..., y_N$. First binarise the data series , and then set the binary matrix z_i, as:

$$z_i = \{1, x_i > mean(y)| - 1, x_i \leq mean(y)\}, i = 1, 2, ..., N.$$

Compute the number of switching in adjacent values in the binary matrix:

$$N_\triangle = \sum_{i=1}^{N-2}\left|\frac{z_{i+1} - z_i}{2}\right|.$$

The Petrosian fractal dimension is calculated as [7]:

$$D = \frac{\log_{10} N}{\log_{10} N + log_{10}(\frac{N}{N+0.4N_{\triangle}})}$$

C. ENTROPY FEATURES

Approximate Entropy: Approximate entropy is quantifies the complexity or irregularity of the signal sequence. Larger the value of it, the more complexity and irregularity of the signal. A robust gauge of approximate entropy can be received by utilising short, noisy datasets.

Sample Entropy:
Larger values means more complexity or irregularity in the EEG data signal. A lower value of SampEn additionally demonstrates more self likeliness in the time series.

Permutation Entropy:
The permutation entropy (PE) portrays the relative occurrence of every pattern. Higher the value of this entropy(nearly 1.0), more the higher frequencies. While the low values of it(nearly 0.4) signifies that the signal contains only low frequencies.

Spectral Entropy:
Spectral Entropy, a standardised type of Shannon's entropy, utilises the power spectrum amplitude segments of the signal sequence for entropy assessment. The spectral entropy(SEN) is computed as:

$$SEN = \sum_{f} p_f \log(1/p_f)$$

where p_f is the power of frequency f. Shannon's Entropy (ShEn) is the estimate of a bunch of relational parameters that changes linearly with the logarithm of the number of potential outcomes. It also estimates EEG signal data spread and is most often used for survey of the dynamical order of a system.

Singular Value Decomposition (SVD) Entropy
The Singular Value Decomposition weights expresses how much of the EEG data set is explained by each vector. SVD entropy estimates richness of feature in the manner that higher the entropy of the arrangement of SVD weights, the more orthogonal vectors are expected to sufficiently explain it.
The following is the obtained normalised values of node features shown for one node(channel) after rounding off as shown in table I:

Kurtosis	0.3089369
Root mean square value	1.293688
Total variation	1.0960908
Hurst exponent	0.245178
Detrended Fluctuation Analysis (DFA)	0.4990422
Hjorth mobility	0.013119
Hjorth complexity	0.228896
Fisher information	0.682297
Higuchi fractal dimension	0.405879
Katz Fractal	0.279842
Petrosian Fractal	0.007783
Sevcik Fractal	0.611357
Approximate entropy	0.66212
Simple entropy	0.69951
SVD entropy	0.79833
Permutation entropy	0.71857
Spectral entropy	0.6095005

TABLE I
NODE LABELLING OF A NODE

VI. GRAPH DESIGNING

The pursuit of specific patterns among the anatomical regions in structural brain MR images is equivalent to sorting out a preferential data flux from a net of nodes belonging to a complete connected graph, being every node a specific anatomical area and every edge a similarity (or dissimilarity) estimate between areas.
We extracted salient brain patterns by calculating the saliency maps:

Each channel of a single data set was converted into a node. We designed a graph g = (V, E), where V denotes a set of nodes and E denotes a set of edges between nodes in V. Data on V can be represented by a feature matrix $A \in R^{n \times n}$, where n = $|V|$ is the number of rows, and d denotes the input feature dimension (here d = 20 features). The edge set E can be represented by a weighted adjacency matrix $A \in R^{n \times n}$ with self-loops, i.e., A_{ii} = 0, where i = 1, 2, ..., n.
Edge weight between the graph nodes g_i^A and $g_j^A$was calculated as [8]:

$$W_A(g_i^A, g_j^A) = d(g_i^A, g_j^A).F(i,j)$$

where $d(g_i^A, g_j^A)$ is the dissimilarity regarding the respective feature data) and F(i, j) represents the spatial closeness between nodes. Dissimilarity is determined such that big feature dissimilarities jump out quickly while alike features have a little effect in the edge weight.

$$d(g_i^A, g_j^A) = \left| \log \frac{T_i}{T_j} \right|$$

Contrarily, closeness is estimated from Graph-based visual saliency(GBVS) algorithm. GBVS is the method which delivers the salient features in an extremely precise and quicker way and in an expounded way. It creates the data in the activation map and afterwards extracts the features from the original picture.

$$F(a,b) = exp\left(\frac{\sum_{i=0}^{d}(a_i - b_i)^2}{2\sigma^2}\right)$$

Feature dissimilarity information is harmonized by the closeness between nodes, therefore encoding graph dissimilarity information at the edges.
Generally, GNNs learn a feature transformation function for X and generates output $Z \in A^{n \times d'}$, where d' signifies the output feature dimension.

VII. NODE EMBEDDING TECHNIQUE

The following discusses the node embedding architecture [4]

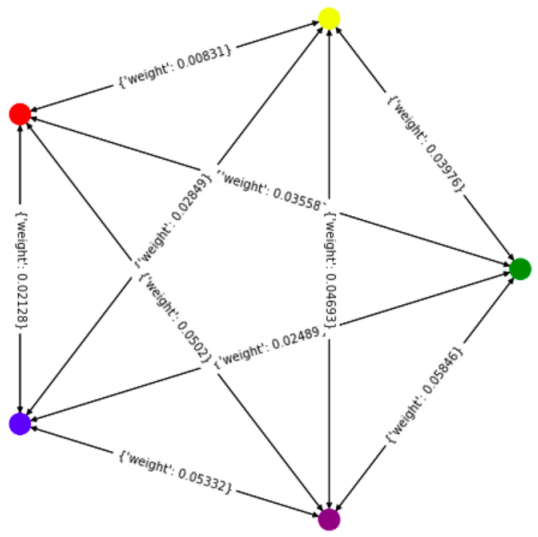


Fig. 3. Sample Graph designing of 5 nodes after their weight calculation

A. Graph Convolutional Networks (GCNs)

SimGNN [4] uses Graph Convolutional Networks (GCNs) [9], which is a neighborhood aggregation based aggregation method to create the graph embeddings. Various other approaches such as DeepWalk [10] and Chebychev [11] for node embedding are also widely popular but is seen that there performance is lower than that of GCNs, because the parameters are different across all computation graphs for each node in each layer and hence the computational cost is more. GCNs are representation invariant and the parameters are shared across are the computational graphs per layer making it computationally cheap. Thus it is a better node embedding technique.

B. Formulation

The present proposed node embedding GCN which utilizes the neighborhood properties for each node via aggregation. Given as:

$$conv(h_n) = ReLu(\sum_{m \in N(n)} \frac{1}{\sqrt{d_n d_m}} h_m W_1^l + b_1^l)$$

Where N(n) is the set of the first-order neighbors of node n plus n itself, d_n is the degree of node n plus 1, $W_1^l \in R^{D^l x D^{l+1}}$ is the weight matrix associated with the l-th GCN layer, $b_1^l \in R^{D^{l+1}}$ is the bias. The learning parameters are shared across all the nodes at each layer and the node representation is position invariant which improves it than other CNN classification methods (LeCun et al., 2015) because it is difficult to define localized convolutional filters and pooling operators, which blocks the transformation of CNN from Euclidean area to non-Euclidean space.

The node features are first one hot encoded and then feeded into the three layered GCN [9]. One hot encoding is permutation invariant so similar nodes have similar one hot encoding. This ensures that aggregation remains the same.

C. Attention Mechanism

In several sequence-based tasks, attention mechanisms have almost become the de facto norm. [12] [13]. Attention makes sure that the nodes which are more important should receive more weights and it is dependent on the specific similarity metric. The attention architecture has a number of intriguing properties:

1) it is compelling since it can be shared across node neighbour pairs,
2) it can be stretched out to graph nodes of different degrees by assigning arbitrary weights to the neighbours; and
3) the model is specifically applicable to inductive learning issues, including activities that require the model to generalise totally.

The node embedding obtained earlier are fed into attention module described as follows:

First a context vector is determined which is simply the normal average of all node embeddings followed by a non-linear transformation

Let

$$U \in R^{NxD}$$

which is the set of all node embeddings

Let $u_n \in U$ denote the n-th row embedding. The context vector $c \in R^D$:

$c = \tanh(\frac{1}{N} W_2 \sum_{n=1}^{N} u_n)$, where $W_2 \in R^{DxD}$ is a learnable weight matrix.

For any node i , attention is calculated by taking the inner cross product of context c and node embedding u_i , given as :

$$a_i = \sigma(u_i^T . c)$$

VIII. Graph Embedding Technique

A. Graph level embedding with global context awareness

The graph embedding is the weighted sum of the attention weights calculated earlier with the node embeddings. Hence the graph embedding h will be given as [4]: $h = \sum_{n=1}^{N} a_n u_n$. In Summary:

$$h = \sum_{n=1}^{N} \sigma(u_n^T c) u_n$$

$$\sum_{n=1}^{N} \sigma(u_n^T c) u_n = \sum_{n=1}^{N} \sigma(u_n^T \tanh(\frac{1}{N} W_2 \sum_{n=1}^{N} u_n)) u_n$$

,where $\sigma(.)$ is sigmoid function.

B. Neural Tensor Networks

The Neural Tensor Network (NTN) [14] replaces a traditional linear neural network layer with a bi-linear tensor layer which connects the two entity vectors directly in multiple dimensions.

Neural tensor networks (NTN) between two graph-level embedding.

Here, a parameter k will manage the number of similarity

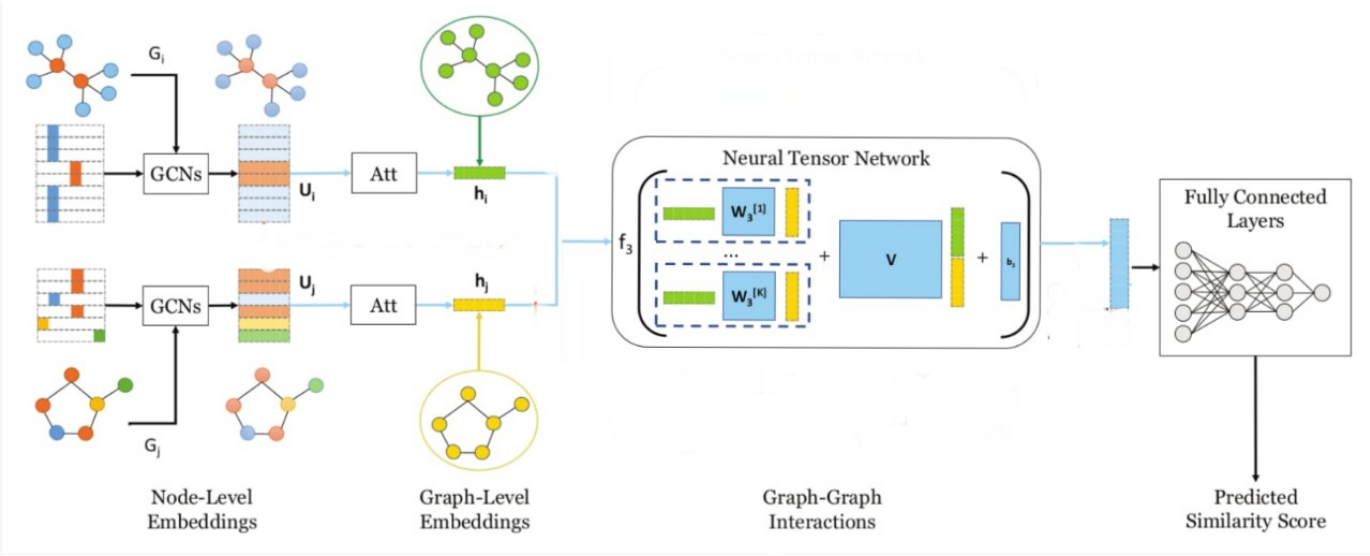


Fig. 4. Figure signifies the graph level embedding data flow, followed by the strategy of Neural Tensor network

scores generated by model for every graph embedding pair. Graph depends upon the weight tensor weight vector and the activation function.

IX. Experimental Results

With the aforementioned BCI competition dataset, extensive experiments are performed to compare the performance of the proposed method with various baseline methods. The performance of our method on each dataset is presented next. BCI competition IV- 2008-IIA is a four-class MI dataset. We here identify each test dataset belonging to a class. The identification can be articulated with the help of similarity score with the SimGNN trained model. Each graph in the validation set is compared with each graph in the training dataset to find the similarity score and thus averaging to find the final prediction score. The lowest test error signifies the the class in which the dataset belongs to. For verification of the similarity in graphs we have used pair-wise Hausdorff Distance computation.

Similarity is inversely proportional to the hausdorff distance found. The similarity score by SimGNN prediction is compared with the with respective pairwise graph hausdorff distance. Though originally Graph Edit Distance (GED) is the measure to find the similarity between two graphs, but due to it's computationally expensive nature, inclusion of GED was not possible.

The model summary on which we trained it on as follows:

A. MODEL SUMMARY

Table II represents model summary for one training set. The graph nodes after channel reduction [6] [15] forms adjacency matrix of (9,9), thus making the process less computationally expensive and efficient. We then proceed with GCN [9] layer and embed features into it. Initializing with 256 neuron units gives the best results. Reduction in units of hidden layer is proceeded with factor of 2 and thus final output to act as input for NTN layer has 32 neuron units thus giving dimension of (9,32). NTN reduced the input to (1,32), and in the final layer we get the final similarity score.

Layer (type)	Output Shape	Param	Connected to
input$_1$($InputLayer$)	[(None, 9, 9)]	0	
input$_2$($InputLayer$)	[(None, 9, 9)]	0	
input$_3$($InputLayer$)	[(None, 9, 9)]	0	
input$_4$($InputLayer$)	[(None, 9, 9)]	0	
graphConv	(None,9, 256)	2560	input$_1$[0][0] input$_2$[0][0] input$_3$[0][0] input$_4$[0][0]
graphConv$_1$	(None,9, 128)	32896	graphConv[0][0] input$_2$[0][0] graphConv[1][0] input$_4$[0][0]
graphConv$_2$	(None, 9, 32)	4128	graphConv$_1$[0][0] input$_2$[0][0] graphConv$_1$[1][0] input$_4$[0][0]
t1(tf.operators.getitem)	(9, 32)	0	graphConv$_2$[0][0]
t2(tf.operators.getitem$_1$)	(9, 32)	0	graphConv$_2$[1][0]
attention (Attention)	(1, 32)	1024	t1[0][0] t2[0][0]
ntn(Neural tensor layer)	(1, 32)	34848	attention[0][0] attention[1][0]
dense (Dense)	(1, 32)	1056	ntn[0][0]
dense$_1$($Dense$)	(1, 16)	528	dense[0][0]
dense$_2$($Dense$)	(1, 8)	136	dense$_1$[0][0]
dense$_3$($Dense$)	(1, 4)	36	dense$_2$[0][0]
dense$_4$($Dense$)	(1, 1)	5	dense$_3$[0][0]
tf.math.sigmoid	(1, 1)	0	dense$_4$[0][0]

TABLE II
MODEL SUMMARY

Total params: 77,217 Trainable params: 77,217

B. IMPLEMENTATION DETAILS

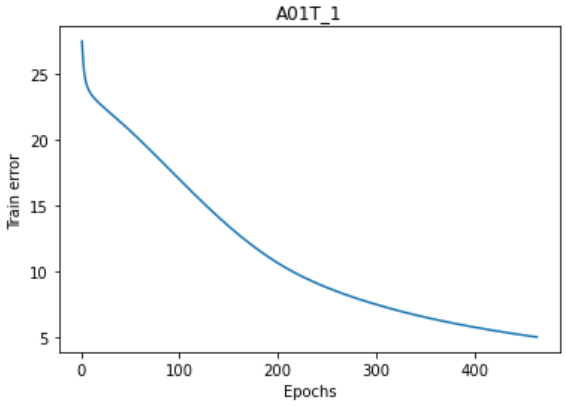
Before beginning the training of model we need to convert a weighted complete graph into a unweighted graph for better training. For that, we get the best results at weight threshold value 0.15.

Input 1 threshold weight:	0.15
Input 3 threshold weight:	0.15
Hausdorff distance:	manhattan
GCN layer 1:	256 nodes
GCN layer 2:	128 nodes
GCN layer 3:	32 nodes
GCN Activation function:	relu
Attention mechanism activation function:	tanh
NTN input dim is:	32
NTN output dim is:	32
Keras dense layers:	16 nodes
Keras dense layers 1:	16 nodes
Keras dense layers 2:	8 nodes
Keras dense layers 3:	4 nodes
Keras dense layers 4:	1 nodes
Dense layer activation func:	relu
Final activation function:	Sigmoid
Learning rate:	0.005
Weight_decay:	10**-4
Train_batch_size:	10
Epochs:	100

TABLE III
MODEL PARAMETERS

C. MODEL TRAINING

Figure 5 shows the value of training errors with progressing epochs. As we can see that with the above model theory and taken parameters values, model converges monotonically and efficiently to provide good performance.



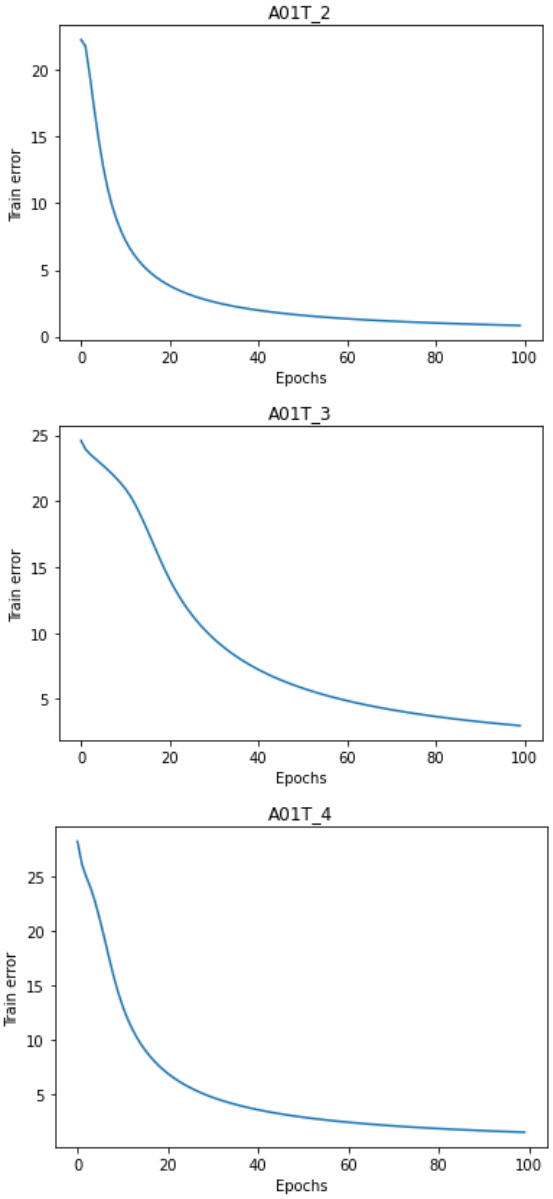


Fig. 5. Train Errors vs epochs

D. PREDICTION TABLE RESULTS

On training the data-set of patient A01 on proposed model. We get the following predictions via identification as shown in Table IV. It can be seen that for the test data of E1 and E4 we get correct predictions from clear margin.

	A01E_1	A01E_2	A01E_3	A01E_4
A01T_1	12.007	26.839	25.46	17.577
A01T_2	14.64	23.391	21.652	22.68
A01T_3	19.714	25.886	22.684	26.334
A01T_4	22.608	22.963	20.078	14.360
Predictions:	**A01T_1**	A01T_4	A01T_4	**A01T_4**

TABLE IV
PREDICTION TABLE

E. PERFORMANCE EVALUATION

We selected 9 channels out of 22 channels overall using MIM (Mutual information maximisation) [15] and results of novel channel selection algorithms [6] on our data-set were compatible to classify T1 and T4 data potentially for patient A01 but because of generalised channel selection results for all patients from novel algorithm, it scarce to produce efficient at few data-sets.

Similarly, on testing the model on other patients data-set. We get the following accuracy as shown in table V.

Patient	Accuracy
A01	2/4
A02	3/4
A03	2/4
A04	2/4
A05	3/4
A06	3/4
A07	2/4
A08	3/4
A09	2/4
Average accuracy:	**61.11**

TABLE V
PROPOSED METHOD PATIENT-WISE ACCURACY

Every patent has the data of 4 classes - training and testing. On the testing data of 4 classes, we identify the correct class it belongs to. Therefore, the accuracy is - number of right identifications/total no. of identifications. Overall we see that an average accuracy of 61.11 percent on identification is achieved.

For each patient/subject we train our model for each class. We form a prediction table and thus evaluate our accuracy. Each cell in table represents test error while training the model. For each row we compare the results with each class and identify our class with the least test error.

Classification Method	**Accuracy**
Decision Tree Classifier	30.83%
K-Nearest Neighbor	31.67%
Support Vector Machine	32.5%
Proposed Methodology	**61.11**%

TABLE VI
IDENTIFICATION ACCURACY

Time complexity of the model on *average takes 77 minutes* to run on every patient. Data-set size is currently 225*4 samples in training, and 225*4 total samples in testing. So overall project runs completely on BCI competition IVa dataset in 11.5 hours. Model is trained on 100 epochs for best results. Now, time evaluation for all single patients to complete predictions individually is shown in the bar graph of Fig. 6.

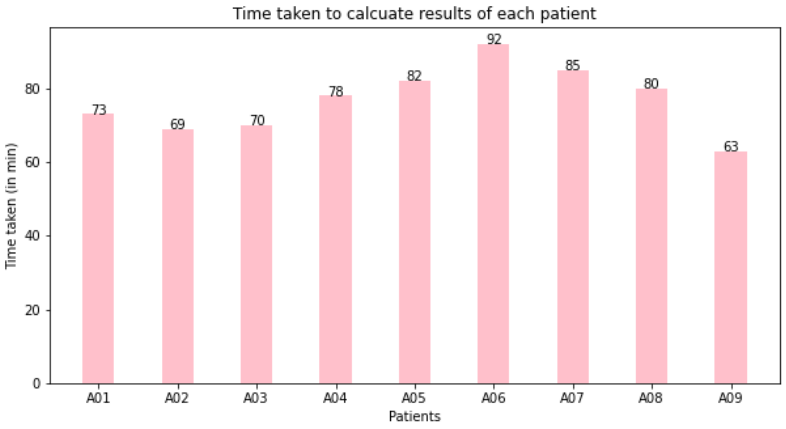


Fig. 6. Time evaluation for all patients

X. CONCLUSIONS

We explored various classification methods like Common Spatial Pattern Recognition, k-NN, SVM, but all them fail to address similarity between various EEG-signals.Whereas Graph Neural Networks approaches were found suitable, where these EEG signals were converted into graph objects. Other Convolution neural network approaches are found to fail over permutation invariant representation of graphs. Whereas, GNN approaches are representation invariant, inductive and learn-able. In the SimGNN [4] approach, the node-node interaction is taken care of by taking global attention awareness, which give more weight-age to those node embeddings which are more important while giving less weight age to less importance nodes for similarity computation. And finally for similarity computation final graph embeddings will be obtained via node embedding aggregation. But it's limited to intra-node attention, when comparing two different graphs one should focus the cross-level interaction between the nodes of different graphs. Also there should be a way to compare the multi granularity between the node embedding of one graph with the graph embedding of another graph. For that further new approaches like Multi Layer Graph Matching Networks [16] where they address node-graph matching network for constructively learning cross-level interactions between nodes of a graph and learn global-level interactions between two input graphs.

The results are promising and can be applied to further classification and identification of various data set. The only challenge is to design suitable graph to be embedded as input in the initial GCN. [9] The representation is to be taken care of, if finding the similarity of different data-sets, then directly the one hot encoded vector input can be used but if the data set is multi-class then the need of finding a feature matrix arises, which is presented in this paper.

Our proposed model works better as compared to decision tree classifier, k-NN classifier and support vector machine as all these classifier are topological based classification, and do not observe the interactions between nodes and graph. Our proposed model trains on node embedding and graph embedding formed which observes node-node, node-graph, graph-graph interactions and thus give feature details between different channels. Our model can be further modified for classification

and can be experimented with other neural network techniques.

XI. Future Work

We can improve our performance by applying Gated Attention Networks(GAaNS) [17]. They use gated aggregation layer for computation of the node-embedding. It controls importance of each attention head by using a small convolutional sub networks. The model's working along with experimental results will be presented in next report. In the paper [7] BP, KNN, Grey relation Analysis, and Random Forest is used separately as classifiers. The recognition rate of the four different classifiers comes around 45 % having low SNR. This recognition rate is very low on poor signal-to-noise ratio with basic classifiers method. That's why we will use a high SNR ratio data (obtained after data pre-processing done in this semester) and will apply Graph neural network methods for classification. The above formulated approach of our paper is a combination of SimGNN and many other approaches that has not been applied on EEG motor imagery signals by any of the referred papers, so predicting the results is a matter of our experiments accuracy.

XII. Acknowledgements

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and our institute. We would like to extend our sincere thanks to all of them. We are highly indebted to our project supervisor Dr. Amrita Chaturvedi for her guidance and constant supervision as well as for providing necessary information regarding the project.

References

[1] M. T. Sadiq, X. Yu, Z. Yuan, and M. Z. Aziz, "Identification of motor and mental imagery eeg in two and multiclass subject-dependent tasks using successive decomposition index," vol. 20, no. 18, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/18/5283

[2] Z. C. S. W. Z. W. L. W. Yufeng Zhang, Xueli Yu, "Every document owns its structure: Inductive text classification via graph neural networks," 2020.

[3] L. Wu, C. Quan, C. Li, Q. Wang, B. Zheng, and X. Luo, "A context-aware user-item representation learning for item recommendation," vol. 37, no. 2. New York, NY, USA: Association for Computing Machinery, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3298988

[4] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang, "Simgnn: A neural network approach to fast graph similarity computation," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, ser. WSDM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 384–392. [Online]. Available: https://doi.org/10.1145/3289600.3290967

[5] N. Jirafe, "How to filter noise with a low pass filter," 2019.

[6] A. Ghaemi, E. Rashedi, A. M. Pourrahimi, M. Kamandar, and F. Rahdari, "Automatic channel selection in eeg signals for classification of left or right hand movement in brain computer interfaces using improved binary gravitation search algorithm," vol. 33, 2017, pp. 109–118. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1746809416302063

[7] C.-T. Shi, "Signal pattern recognition based on fractal features and machine learning," 2018, pp. 3–8.

[8] A. Rueda, F. A. González, and E. Romero, "Extracting salient brain patterns for imaging-based classification of neurodegenerative diseases," vol. 33, no. 6, 2014, pp. 1262–1274.

[10] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk." ACM, Aug 2014. [Online]. Available: http://dx.doi.org/10.1145/2623330.2623732

[11] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 3844–3852.

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016.

[13] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, "A convolutional encoder model for neural machine translation," 2017.

[14] D. Chen, R. Socher, C. D. Manning, and A. Y. Ng, "Learning new facts from knowledge bases with neural tensor networks and semantic word vectors," 2013.

[15] M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic, "On mutual information maximization for representation learning," 2020.

[16] X. Ling, L. Wu, S. Wang, T. Ma, F. Xu, A. X. Liu, C. Wu, and S. Ji, "Multi-level graph matching networks for deep graph similarity learning," 2021.

[17] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "Gaan: Gated attention networks for learning on large and spatiotemporal graphs," 2018.