Learning Brain Effective Connectivity from EEG data

by

Amar Enkhbat

A Master Thesis

Submitted to
the Graduate School of the Tokyo Institute of Technology
on January 24$^{\text{th}}$, 2022

Thesis Supervisor: Tsuyoshi Murata
Professor of Computer Science

0.1 Introduction

Motor movement imagery classification is one of the most widely studied task in the field of Brain Computer Interface (BCI). Creating a BCI that can accurately classify motor movement imagery has numerous potential applications such as: creating interface between computers and humans with partial or full motor disability, supporting motor rehabilitation of post-stroke patients, video game controller and various other neural interface devices. This is usually accomplished by 1) acquiring brain signals, 2) Extracting features, 3) Decoding features and finally 4) Outputting commands. Common ways of acquiring brain signals are electroencephalography (EEG) and functional Magnetic Resonance Imaging (fMRI). Even though fMRI is able to produce both temporally and spatially higher resolution data than EEG, EEG's accessibility, compactness and affordability is preferred to fMRI by most researchers in the field.

There are 2 widely used paradigms for BCI. First paradigm is mostly found in traditional BCI where brain signals are preprocessed using signal processing methods such as Fourier transforms, band-pass filters, spatial filters to extract frequency domain, band power, common spatial pattern (CSP) features. These extracted features are decoded using classifiers such as LDA, SVM, Gaussian classifier etc. Second paradigm is end-to-end Deep Learning (DL) where brain signals are handled as time-series data and decoded using Deep Neural Networks (DNN) with complex architecture and large amount of trainable parameters. These models are usually composed of several hidden convolutional layers, recurrent layers and attention layers connected sequentially or in parallel. These models have shown to frequently outperform traditional BCI methods. Even though both paradigms are widely studied, neither takes into account that the brain is a inherently a huge interconnected network with various neurons connected to other neurons through synapses. Various studies show that specific regions of the brain are specialized for specific task and these regions show increased electric activity when said task is being conducted. The work on this paper focuses on this fact.

The paradigm of this paper treats the brain as a network or a graph. Recent work showed that by manually predefining graph structure of the brain they can convert time-series signals into series of temporal graphs. Subsequently, they used these features as input and applying 1-hop message passing using the adjacency matrix of the graph and applying convolutional layers, recurrent layers and an attention layer they were able to achieve state-of-the art results on PhysioNet Motor Movement Imagery (MMI) dataset and BCI 2a dataset. Even though this method used graph as features they used conventional neural networks instead of Graph Neural Networks (GNN) that are more suited to graph structured dataset. Second, their work predefined connectivity or edge for graph conversion. Even with domain knowledge in fields such as neuroscience or cognitive science etc this task is extremely difficult because we still lack sufficient information about the neural connections in the brain. Even if we did have such information its likely that brain connections could differ depending on age, gender, mother language, profession etc. Previous works used basic methods such as connecting neighboring nodes or using Euclidean distance as edge features however results in this paper show that these methods are not optimal.

In this work, we propose a modified Graph Convolutional Network (GCN) that can simultaneously learn connectivity of the brain and classify motor movement imagery. In this work, instead of using predefined adjacency matrix, we propose turning the adjacency matrix into a learnable parameter $A \in R^{n \times n}$ where n is the number of EEG nodes. Before training the adjacency matrix it is initialized with

Glorot initialization method then its diagonal is initialized as 1. After training, A represents optimal connections of EEG nodes for the trained task.

The key contributions are summarized as follows:

1. We proposed a modified GCN called *AutoGCN* which has the ability to learn optimal connectivity of a graph for the training task and classify motor movement imagery simultaneously.

2. The theoretical expressive power of *AutoGCN* is proven to show that it can find more optimal adjacency matrix compared to predefined adjacency matrices in previous works

3. Experiments and ablation studies are conducted to further explore graphs learned by *AutoGCN*

4. Propose further improvements for *AutoGCN*

0.2 Background

This chapter will provide necessary background information for this research by defining problems, explaining related works, intended tasks and their advantages and disadvantages.

0.3 Graphs

A graph is a data structure that shows relationship between set of objects by connecting pair of objects with edges. The mathematical definition of a graph is as follows:

Let $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be a simple undirected graph. Then $\mathcal{V} = \{1, ..., |\mathcal{V}|\}$ is the set of $N = |\mathcal{V}|$ number of vertices or nodes, $\mathcal{E} \subseteq \{(u, v)|u, v \in \mathcal{V}\}$ is set of pair of edges and $\mathbf{X} \in \mathbb{R}^{N \times d}$ is matrix of d-dimensional node features. Adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times N}$ can be used to represent connections between nodes $\mathcal{V}$ where $\mathbf{A}_{ij} = 1$ if $(i, j) \in \mathcal{E}$ otherwise 0. $\mathbf{A}_{ij}$ can have values other than binary then the graph G becomes weighted graph. Also, if adjacency matrix $\mathbf{A}$ is symmetric or $\mathbf{A} = \mathbf{A}^T$ the graph G is called an undirected graph. If $\mathbf{A} \neq \mathbf{A}^T$ then it's called a directed graph. Graphs can be used to model many types of relations and processes both physical and abstract. For example, social networks are graphs where people nodes and their social interactions are edges. Also, graphs are also called networks.

0.4 Graph Convolutional Networks (GCN)

Graph neural networks are type of neural network that is specifically designed to work on graph-structured data and take advantage of their properties. GNNs have gained popularity recently thanks to the contribution of Kipf et al and their Graph Convolutional Networks. Since then, many papers on GNNs have been published such as GAT, GraphSAGE, GraphSAINT etc. However GCNs are considered as one of the basic GNN variant and even tho7ugh there are technically 2 variants of GCNs: Spatial GCN and Spectral GCN we will use GCN as defined by Kipf et al.

A GCN layer is defined as follows:

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \tag{1}$$

Abstract

Brain-computer interface or BCI is a direct communication pathway between the brain's electrical activity and and external electronic devices such as computers or robotic wheelchair or limbs. Creating a BCI that can accurately classify motor movement imagery has numerous potential applications such as: creating interface between computers and humans with partial or full motor disability, supporting motor rehabilitation of post-stroke patients, video game controller and various otherneural interface devices. This is usually accomplished by 1) acquiring brain signals, 2) Extracting features, 3) Decoding features and 4) Outputting commands. There are 2 widely used paradigms for BCI. First paradigm is traditional BCI methods where brain signals are preprocessed using signal processing methods such as Fourier transforms, band-pass filters, spatial filters to extract frequency domain, band power and common spatial pattern (CSP) features. Then extracted features are decoded using classifiers such as LDA, SVM, Gaussian classifier. Second paradigm is end-to-end Deep Learning (DL) where brain signals are handled as time-series data and decoded using Deep Neural Networks (DNN) with complex architecture and large number of trainable parameters. Even though both are widely studied, neither paradigm consider the inherent interconnected graph-like structure of the human brain. In this work we propose a graph neural networks model called AutoGCN which can simultaneously learn graph structure of EEG nodes and classify motor movement imagery. AutoGCN accomplishes this by:

1. Using trainable adjacency matrices to represent structure
2. Calculating graph encodings using stacked Batch Graph Convolution layers.

Batch Graph Convolution layer is a modified version of Graph Convolution layer originally proposed by Kipf et al. in 2016. Key difference between a BatchGraphConv layer and the original GraphConv layer is that BatchGraphConv can handle multiple, or batch of graphs as opposed to GraphConv which can handle only a single graph. Finally, experiments conducted in Physionet Motor Movement Imagery dataset show that AutoGCN can learn more optimal adjacency matrix than previous works' manually defined adjacency matrices. Also, we show that previous work augmented by our method achieves state-of-the-art results in motor movement imagery tasks. Furthermore, we shot that our method can be used other data than EEG by using MNIST as an example.

Contents

List of Figures

List of Tables

1 Introduction

Motor movement imagery classification is one of the most widely studied task in the field of Brain Computer Interface (BCI). Creating a BCI that can accurately classify motor movement imagery has numerous potential applications such as: creating interface between computers and humans with partial or full motor disability, supporting motor rehabilitation of post-stroke patients, video game controller and various other neural interface devices. This is usually accomplished by 1) Acquiring brain signals, 2) Extracting features, 3) Decoding features and 4) Outputting commands. Two most common ways of acquiring brain signals are electroencephalography (EEG) and functional Magnetic Resonance Imaging (fMRI). Even though data from EEG have both temporally and spatially lower resolution data, its accessibility, compactness and affordability is preferred over fMRI by most researchers in the field.

There are two widely used paradigms for BCI. First paradigm is mostly found in traditional BCI where brain signals are preprocessed using signal processing methods such as Fourier transforms, band-pass filters, spatial filters to extract frequency domain, band power and common spatial pattern (CSP) features then decode these features using classifiers such as LDA, SVM, Gaussian classifier. Second paradigm is end-to-end Deep Learning (DL) where brain signals are handled as time-series data and decoded using Deep Neural Networks (DNN) with complex architecture and large amount of trainable parameters. These models are usually composed of several hidden convolutional layers, recurrent layers and attention layers connected sequentially or in parallel and stacked on top of each other. Furthermore, these models have shown to frequently outperform traditional BCI methods. Even though both paradigms are widely studied, neither takes into account that the brain is inherently a huge interconnected network with various neurons connected to each other through synapses. We knw that some regions of the brain are specialized for specific task and these regions show increased electric activity when a task is being conducted [7]. This paper explores a computational way to leverage brain connectivity to increase further BCI performance. The paradigm of this paper treats the brain as a network or a graph.

Recent works [18] have shown that by manually defining graph structure of the brain or more specifically graph structure of EEG electrodes, they can convert time-series signals into series of temporal graphs. They used these tempral graphs as features and by applying 1-hop message passing and applying convolutional layers, recurrent layers and an attention layer they were able to achieve state-of-the art results on PhysioNet Motor Movement Imagery (MMI) dataset [13]. However this method has two shortcomings:

1. Even though this method used temporal graphs as features they used conventional neural networks instead of Graph Neural Networks (GNN) that are more suited to graph structured dataset.

2. They manually defined the adjacency matrix of the graph using Euclidean distance between EEG electrodes which is not guaranteed to be optimal graph structure.

In this work, we propose a GNN model that is based on Graph Convolution Network (GCN)[8] called AutoGCN. AutoGCN can simultaneously learn effective connectivity of the EEG electrodes and classify motor movement imagery. This is accomplished by 2 features of AutoGCN:

1. Instead of using manually defined adjacency matrices, we propose to use trainable adjacency matrix $\mathbf{W_A} \in \mathbb{R}^{N \times N}$ where N is the number of EEG electrodes. Before training $\mathbf{W_A}$ is initialized with Glorot initialization as described in [4]. Then its diagonal is initialized as 1 in order to conserve node attributes initial self connections.

After training, $\tilde{\mathbf{W}}_{\mathbf{A}} \in \mathbb{R}^{N \times N}$ represents optimal connections of EEG electrodes for the trained task and subsequently could be used to visualize effective connectivity of the EEG electrodes.

The key contributions of this work are as follows:

1. We proposed a modified GCN called AutoGCN which takes time-series graphs as input and has the ability to learn optimal adjacency matrix for Motor Movement Imagery classification.
2. We show that that AutoGCN can find more optimal adjacency matrix compared to manually defined adjacency matrices from previous work
3. Experimental studies are conducted to further explore adjacency matrix learned by AutoGCN
4. Propose further improvements for AutoGCN

2 Background

This section will provide necessary background information for this research by defining problems, intended tasks and explaining related works and their advantages and disadvantages.

2.1 Graphs

A graph is a data structure that shows relationship between set of objects by connecting pair of objects with edges. Mathematical definition of a graph is as follows:

Let $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be a simple undirected graph. Then $\mathcal{V} = \{1, ..., N\}$ is the set of N number of vertices or nodes, $\mathcal{E} \subseteq \{(u, v)|u, v \in \mathcal{V}\}$ is set of pair of edges and node attributes $\mathbf{X} \in \mathbb{R}^{N \times d}$ is N number of samples for each node with d-dimensional features. An adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times N}$ can be used to represent connections between nodes $\mathcal{V}$ where $A_{ij} = 0$ if $(i, j) \notin \mathcal{E}$ otherwise it can have any real value. Adjacency matrix is preferred to node pair lists because it can show properties of the grahp. For example, if $\mathbf{A} = \mathbf{A}^T$, where $\mathbf{A}^T$ is transpose of $\mathbf{A}$, then the graph is symmetric or an undirected graph. If values of $\mathbf{A}_{ij}$ are binary then its a unweighted graph. If a adjacency matrix is neither symmetric nor its values are binary then its a directed and weighted graph.

2.2 Graph Neural Networks (GNN)

GNNs are type of neural network that are specifically designed to work with graph-structured data. GNNs have gained popularity recently thanks to the contribution of Kipf et al and their Graph Convolutional Networks (GCN). Even though research on GNNs were rapid most GNNs such as GAT[16], GraphSAGE[5], GraphSAINT[17] etc are heavily based on GCN. GCN is considered the basic building block of GNNs similar to what Fully Connected Networks (FCNs) are to regular DL models. Even though there are different variants of Graph Convolution (GC) layers, we will use fast approximate convolutions on graphs originally defined by Kipf et al. as a GC layer.

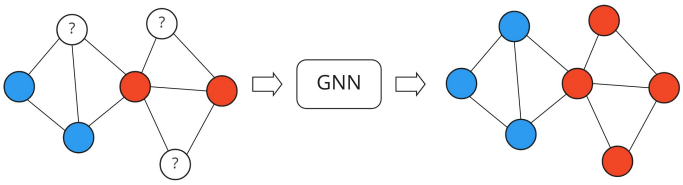
A single **GC** layer is defined as follows:

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \tag{1}$$

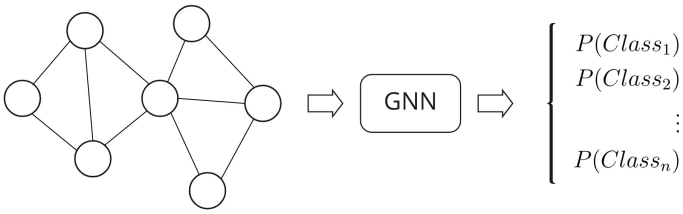
where, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ is the adjacency matrix of the graph G with added self-connections. $\mathbf{I}_N$ is the identity matrix, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$, and $\mathbf{W}^{(l)}$ is a layer-specific trainable weight matrix. $\sigma(\cdot)$ denotes an activation function, such as the $ReLU(\cdot) = max(0, \cdot)$. $\mathbf{H}^{(l)} \in \mathbb{R}^{N \times D}$ is the matrix of activations in the l^{th} layer; $\mathbf{H}^{(0)} = \mathbf{X}$.

A GC layer is very similar to the basic FC layer except added matrix multiplication of $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ in front of the output of previous layer $\mathbf{H}^{(l)}$ and its trainable weights $\mathbf{W}^{(l)}$. This multiplication allows the GC layer to aggregate node's attributes and its neighboring nodes attributes into new encoding. This operation is called **message passing and aggregation**. One GCN layer allows us to compute 1-hop neighbor aggregation. By stacking k number of GCNs we can compute k-hop message aggregation between nodes.

There exists wide range of graph-related tasks such as node classification/regression, graph classification/regression, community detection, link prediction, influence prediction/maximization, graph generation etc. However, the work on this paper will focus on primarily on graph classification. The difference between node classification task and graph classification task is shown in Figure 1.



(a) Node classification task



(b) Graph classification task

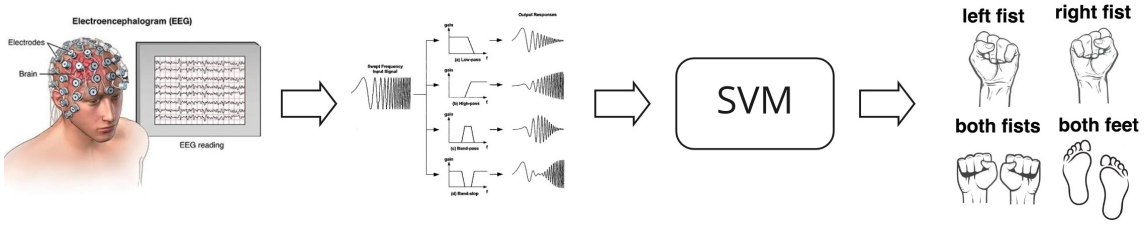
Figure 1: *Top*: In node classification task a GNN is trained subset of of nodes. During prediction phase GNN classifies unknown from the same graph. *Bottom*: In graph classification task a GNN is trained with series of graphs and their labels. During prediction phase GNN classifies unseen graphs' labels.

2.3 Brain Computer Interface

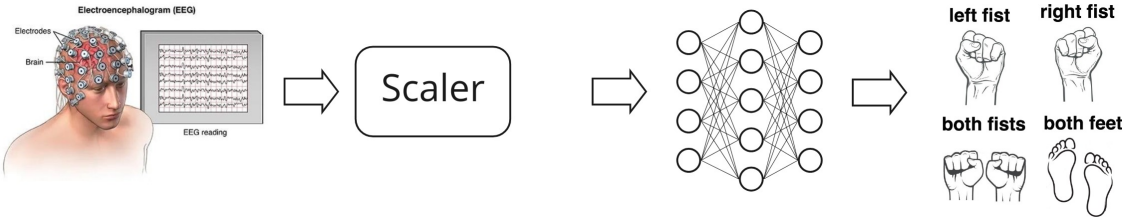
A Brain Computer Interface (BCI) is a direct communication interface between the brain's electrical activity and an external device. This is accomplished by 1) acquiring brain signals, 2) Extracting features, 3) Decoding features and 4) Outputting commands. There are various applications for BCIs such as interface between computers and humans with partial or full motor disability, supporting motor rehabilitation of post-stroke patients, video game controller and connection with various other neural interface devices. These applications are usually achieved through intention classification BCIs. An intention classification BCI is a BCI that can classify a subject's intent. In other words intention classification BCIs decode what a subject is thinking using brain activity alone. Motor Movement Imagery (MMI) classification is especially useful for these types of tasks. In MMI classification task, a BCI classifies subjects' imagined motor movement function such as raising an arm up or down, clenching or releasing fist, moving tongue etc. MMI BCI is especially useful for persons with partial or full motor disability because MMI BCI allows for additional way to interact with electronic devices such as electronic wheelchair, robotic arms, smartphones and other devices. Also, recent works [11][2] show that BCIs can be used with regular rehabilitation program to restore motor movement and speech abilities for post-stroke patients. Furthermore, MMI BCIs are not limited to medical use and can be used for entertainment purposes as well. Common MMI BCI pipeline is shown in Figure 2.

There are multiple methods to acquire brain signal data. Two most common methods are electroencephelography (EEG) and functional Magnetic Resonance Imaging (fMRI). In fMRI, subject's electrical brain signals are recorded using fMRI machine. Usually this method produces both spatially and temporally higher resolution brain signal recordings than EEG. However, due to the size and cost of fMRI machines this method is usually used for recording static subjects. Conversely, EEG recording devices are accessible, compact and affordable making it ideal for BCI.

When recording brain activity using EEG, a subject wears a cap with 16-256 electrodes with fixed positions. These electrodes record potential difference between itself and the reference electrode. The potential differences represent how much activity is happening around the electrode. Due to its compact size, portability and affordability EEG recorders



(a) Traditional BCI pipeline



(b) Deep learning BCI pipeline

Figure 2: *Top*: In traditional BCI each EEG signal is treated like a signal and features are extracted using signal processing methods. Extracted features are decoded using classifiers such as SVM. *Bottom*: In DL based BCI methods preprocessing methods like standard or MinMax scaling is used to scale the EEG data then fed into a DNN classifiers of various architectures.

are preferred by BCI researchers, however due to electrical activity around the scalp, hair and due to the fact that EEG captures only surface level potential differences it produces heavily noisy data.

There are 2 widely used paradigms for BCI which are shown in Figure 2. First paradigm is mostly found in traditional BCI where brain signals are preprocessed using signal processing methods such as Fourier transforms, band-pass filters, spatial filters to extract frequency domain, band power, common spatial pattern (CSP) features. These extracted features are decoded using classifiers such as LDA, SVM, Gaussian classifier. Second paradigm is end-to-end DL based methods where brain signals are handled as time-series data and decoded using Deep Neural Networks (DNN) with complex architecture and large amount of trainable parameters. These models are usually composed of several hidden convolutional layers, recurrent layers and attention layers connected sequentially or in parallel. DL models have shown to frequently outperform traditional BCI methods by a large margin and does not require domain knowledge in neuroscience and cognitive science.

2.4 The brain as a graph

There are two ways where we can model the brain as a graph.

1. First, we can model the brain into a graph using its physical connections:
   Let $G_{brain} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$. Then its nodes $\mathcal{V}$ becomes brain's neurons and $\mathcal{E}$ represents the synapses that connect neurons to each other.

2. Second, we can model the brain as a graph by analyzing its signals. When recording brain signals using EEG, its electrodes are placed in a manner so that it can capture as much area of the brain as possible. Because EEG electrodes are placed outside the brain, its recordings represent generalized brain activity of the position it is placed on. And the influence that a EEG node exerts over another is called **effective connectivity**. This is called *effectve* connectivity because nodes' influence each other does not necessarily depend on their physical connections of the brain. We

can define this graph structure as: Let $G_{brain} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$. Then its nodes $\mathcal{V}$ becomes EEG bides and $\mathcal{E}$ becomes set of weighted edges that show how much a node pair influence each other.

3 Related works

This section gives examples of MMI classification BCI methods, their advantages and disadvantages.

3.1 Traditional methods

Common Spatial Pattern (CSP)[9] is one of the most popular and effective feature extraction methods in MMI classification. It is a spatial filtering approach that tries to find a linear combination of EEG channels that the power difference of different motor imagery classes is maximized. There have been numerous works that extend CSP and achieve remarkable improvement. One of its most successful variants is the Filter Bank CSP (FBCSP)[1], which addresses CSP's drawback of depending on a particular frequency band by applying CSP to different frquency bands and selecting subject-specific features by a feature selection method. Until recenty, FBCSP has been state-of-the-art method in EEG classification and has provided good results. In terms of classifiers, traditional algorithms like SVM and LDA are commonly used in many EEG MMI tasks with CSP, FBCSP and their variants.

3.2 DL models

Recently, DL models have started show excellent results in tasks that were originally thought to be extremely hard for computers. Convolutional Neural Networks [10], which is based on how the human brain analyzes visual imagery, have shown amazing results in various tasks dealing with images such as image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, and even some sequence data such as time-series data and natural language processing. Recurrent Neural Networks, which is derived from FCN where connections between nodes form a directed graph along a temporal sequence, showed that it can handle various sequential data such as temporal or time-series data, text data and learn temporal dependencies. Improved versions of RNNs[12] such as LSTM[6], GRU[3] and Transformers[15] also showed even better results in sequential data such as time-series data and text, speech data.

DL models have also being used for various BCI tasks. Schirrmeister et al.[14] proposed a CNN BCI model and achieved better performance than FBCSP. Recent works by Dalin Zhang et al. [20][19][21] have proposed various end-to-end DL models that all outperform traditional methods and their previous models. The Cascade and Parallel Convolutional Recurrent Neural Networks (CRNN) both proposed in [20] which incorprorated both CNN and RNNs showed impressive accuracy in classifying MMI from EEG dataset. Specifically, Cascade CRNN's convolutional layers extracted spatial features from EEG nodes by treating the EEG node positions as series of images and its recurrent layers learned temporal dynamic between extracted convolutional kernels then its outputs are fed into FCN layers and Softmax function to compute final class probabilities. In Parallel CRNN, its convolutional and recurrent layers learn from time-series EEG data directly and their outputs are concatenated. Then concatenated outputs are fed into FCN layers and softmax layer to compute final probabilities. Both Cascade and Parallel CRNN achieved significantly higher accuracy compared to traditional methods such as FBCSP. Furthermore, results from these works showed that both spatial and temporal features are important for MMI classification task.

3.3 Previous work: G-CRAM

Even though DNN based methods provide state-of-the-art results, there have been only handful of methods that make use of the fact that the human brain which is a biological

neural network composed of billions of neurons connected through synapses. If we consider brain neurons as nodes of a graph and its synapses edges, we can treat the human brain as a graph or a network. Even though we knew still know little about how these neurons and connections are biologically created, we do know that certain parts of the brain are dedicated to certain tasks. Making use of this fact could potentially improve accuracy of DNN models even further.

Another model proposed by Zhang et al.[18] called Graph-based Convolutional Recurrent Attention Model (G-CRAM) make use of graph structure of the human brain. They manually defined a graph structure based on physical distance between EEG nodes. They proposed 3 different ways to construct a graph: N-graph, D-graph and S-graph.

1. N-graph.

   Define a graph $G = (\mathcal{V}, \mathcal{E}_N, \mathbf{X})$, then $\mathcal{V} = \{s^i | i \in [1, N]\}$ is EEG nodes $\mathcal{E}_N = \{s^i s^j | (i,j) \in \mathbf{H}\}$ where $\mathbf{H}$ and $\mathbf{X}$ is signal values. In other words N-graph simply connects neighboring EEG nodes then assigns its signal values as node attributes. Its adjacency matrix can be computed as

   $$\mathbf{A}_N = \begin{cases} 1 & \text{if } s^i s^j \in \mathcal{E}_N \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

   Figure 5a shows the adjacency matrix and constructed graph structure of N-Graph.

2. D-graph and S-graph

   In D-graph and S-graph nodes and their attributes are same as N-graph but their edges are defined differently. Euclidean distance between EEG nodes are used to connect nodes in the graph. For D-graph and S-graph, its adjacency matrices $\mathbf{A}_D$ and $\mathbf{A}_S$ are defined as:

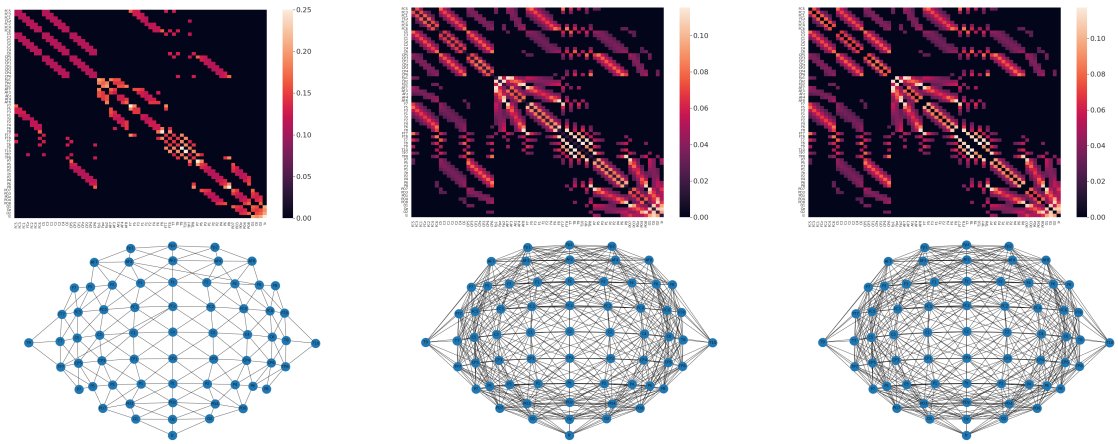
   $$\mathbf{A}_D = \begin{cases} \frac{1}{d_{ij}} & \text{if } d_{ij} < \mathbf{E}(L) \\ 0 & \text{if } d_{ij} \geq \mathbf{E}(L) \\ \frac{1}{\mathbf{E}()\{d_{iq} | d_{iq} < \mathbf{E}(L), q \in [1,n]\}} & \text{if } i = j \end{cases} \tag{3}$$

   $$\mathbf{A}_S = \begin{cases} \frac{1}{d_{ij}} & \text{if } d_{ij} < \mathbf{E}(L) \\ 0 & \text{if } d_{ij} \geq \mathbf{E}(L) \\ \frac{1}{Min(d_{iq} | q \in [1,n])} & \text{if } i = j \end{cases} \tag{4}$$

   respectively, where $\mathbf{E}(L)$ is the average of distance set L. In other words, if a EEG node pair Euclidean distance is above the average, they're considered not connected. If it is below average, its edge values is equal to inverse of the Euclidean distance. Edge connections of D-graph and S-graph are exactly same only diagonal value of their adjacency matrices are different. Figures 5b and 5c show visualization of matrices and graphs of D-graph and S-graph respectively.

3.4 Drawbacks of previous works

Even though G-CRAM showed state-of-the-art results on MMI EEG task, its physical distance based graph construction method is overly simplified and not optimal. Even though the human brain is likely to be connected based on physical distance, its effective connectivity does not necessarily follow this trend. Recent works showed that even though some parts of the brain are specialized in some tasks, recordings show that brain activities overlap with various other areas of the brain. In other words, the brain sends various



(a) N-graph (b) D-graph (c) S-graph

Figure 3: Manually defined graph structure from G-CRAM. *Top row* Heatmaps of each graph's adjacency matrices. *Bottom row* Constructed graphs using the adjacency matrices.

signals all around the human brain when doing specific tasks which do not necessarily follow the traditional convention that some physical parts of the brain are specialized for specific task. Furthermore, previous work assumes that brain functions similar in every humans. Brain effective connectivity could be different depending on the subject's age, gender, ethnicity, experience, profession etc and it is naive to assume that a single graph structure can represent all types of people's graphs. And finally, G-CRAM used only single 1-hop message passing to get graph encodings $\mathbf{Z} = \mathbf{AX}$. Then the graph encoding $\mathbf{Z}$ is fed into series of convolutional, recurrent layers and attention layers. Similar to the physical distance problems this also assumes that brain regions only interact with neighboring regions. It could be beneficial to consider 3-hop message passing and aggregation between EEG in order to improve accuracy. Drawback of previous works are summarized as follows:

1. Traditional methods require domain knowledge in neuroscience, cognitive science etc
2. DL based models do not take into account graph nature of the brain
3. G-CRAM model manually defined overly simple connections for representing EEG data as graphs
4. G-CRAM only aggregates node attributes from only 1-hop neighbors.

3.5 Research target

The target of this research is to propose a EEG MMI classification model that that address problems defined previously. Because we do not have domain knowledge in neuroscience or cognitive science it is decided that we use DL based approach to MMI classification. Also, in order to address the EEG data to temporal graph conversion problem we decided to let adjacency matrix a learnable parameter instead of manually defining which also requires domain knowledge and train simultaneuosly with the model. And finally, to address 1-hop message passing and aggregation, we decided to use multiple GC layers to compute k-hop message passing and aggregation of EEG nodes.

We define the problem of this paper as follows:

Definition 1 (**Supervised Effective Connectivity Prediction**)**.** Let $\mathbf{X} \in \mathbb{R}^{m \times n}$ be a set of samples, $\mathbf{Y} \in \mathbb{R}^{m}$ its corresponding targets where where m is number of samples and n number of features. Additionally we define a learnable matrix $\mathbf{W_A}$. Then the task of

finding optimal effective connectivity or optimal adjacency matrix $\tilde{\mathbf{W}}_{\mathbf{A}}$ by training $\mathbf{W}_{\mathbf{A}}$ using model $\mathcal{F}$ where $\mathcal{F}(\mathbf{X}, \tilde{\mathbf{W}}_{\mathbf{A}}) = \mathbf{Y}$ will be called Supervised Effective Connectivity Prediction.

4 AutoGCN

In this paper we propose a modified version of GCN called AutoGCN which takes series of graphs $G_t = (\mathcal{V}, \mathcal{E}, \mathbf{X}_t)$, with shared randomly initialized edges, as inputs and outputs their graph label probabilities $P(\mathbf{Y}_t)$ and their optimal adjacency matrix $\tilde{\mathbf{W}}_\mathbf{A}$. AutoGCN differs from regular GCN in two main ways. First, AutoGCN takes trainable adjacency matrix $\mathbf{W}_\mathbf{A}$ instead of predefined adjacency matrices. Second, AutoGCN can work with series of graphs instead of a single graph. These two modifications allow AutoGCN to take series of graphs converted from EEG data is input and output their class probabilities. The two modifications are explained in detail below Sections 4.1 and 4.2.

4.1 Trainable adjacency matrix

In order to predict effective connectivity of EEG nodes, instead of using manually defined adjacency matrices similar to G-CRAM, we propose to traina adjacency matrix $\mathbf{W}_\mathbf{A} \in \mathbb{R}^{N \times N}$. After training AutoGCN on EEG data, $\tilde{\mathbf{W}}_\mathbf{A}$ represents the optimal effective connectivity between EEG nodes and can be used to visualize effective connectivity between nodes and can be used for EEG MMI classification tasks. In other words, in addition to AutoGCN's own parameters $\mathbf{W}$, it also trains adjancency matrix $\mathbf{W}_\mathbf{A}$ at the same time using backpropagation.

Early in our research we decided to use DL equivalent of GCN renormalization technique in first AutoGCN layer. Instead of using GC layer computation

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{W}_\mathbf{A}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \tag{5}$$

we initially used

$$\mathbf{H}^{(l+1)} = \sigma(Softmax(\mathbf{W}_\mathbf{A}\mathbf{W}_\mathbf{A}^T + \mathbf{I}_N)\mathbf{H}^{(l)}\mathbf{W}^{(l)}) \tag{6}$$

to compute AutoGCN layers. Intuition behind this was that $\mathbf{W}_\mathbf{A}\mathbf{W}_\mathbf{A}^T$ is a symmetric matrix therefore represents undirected graph, addition of identity matrix $\mathbf{I}_N$ to represents self connections of EEG nodes and $Softmax(\cdot)$ function was used for normalization similar to renormalization trick of GCN.

However, during testing we found out that this method did provide optimal results for motor movement classification from EEG data. Instead, we found that by simply initializing $\mathbf{W}_\mathbf{A}$ with Glorot initialization then filling its diagonal values with 1, $diag(\mathbf{W}_\mathbf{A}) = 1$, AutoGCN showed significantly better than using (6). Detailed comparison of computation on adjacency matrix will be shown in Section 5. In our final model $\mathbf{W}_\mathbf{A}$ was initialization following these two steps:

1. Initialize $\mathbf{W}_\mathbf{A}$ with Glorot initialization.
2. Fill diagonal values with 1.

4.2 Batch Graph Convolution layer

Regular graph convolution layers only work with node classification tasks and are unable to compute graph convolution of multiple graphs. A single graph convolution layer takes subset of nodes from a graph G then predicts unknown nodes' labels of the same graph. For EEG signals, electrode values of each time step m is considered a separate graph $G_m = (\mathcal{V}, \mathcal{E}, \mathbf{X}_m)$ where $m \in \{1, .., M\}$ and M is number of batches. During training, graphs are separated into graphs: $\mathbf{G}_{train}$, $\mathbf{G}_{valid}$ and $\mathbf{G}_{test}$. Usual approach for computing batch of graphs using graph convolution is accomplished by diagonally creating a $\mathbf{A} \in \mathbb{R}^{MN \times MN}$ by stacking adjacency matrices $\mathbf{A} \in \mathbb{R}^{N \times N}$ however this method requires large amount of

memory and needlessly complex. Instead we simply use Einstein notation and PyTorch's built in torch.einsum() function to compute each graph's encodings independently.

Computation of a Batch Graph Convolution layer is formulated as follows:

Given a batch of graphs $\mathbf{G} = \{G_1, G_2, ..., G_M\}$ where $M = |\mathbf{G}|$ is the batch size with accompanying node attributes $\mathbf{X}_{batch} = \{\mathbf{X}_i, \mathbf{X}_{i+1}, ..., \mathbf{X}_{i+M}\} \in \mathbb{R}^{M \times N \times d}$, Batch Graph Convolution layer computes encoding of each graph separately using following function:

$$\mathbf{H}_{iml}^{(l+1)} = \sigma(\sum_k \sum_j \mathbf{W}_{\mathbf{A}mj} \mathbf{X}_{ijk} \mathbf{W}_{kl}) \tag{7}$$

where $\mathbf{H}_{iml}^{(l+1)}$ is Batch Graph Convolution Layer's encoding at i-th sample, m-th node and l-th feature.

This computation method does not require to stack adjacency matrix on top of each other and can be implemented simply using PyTorch library.

4.3 Overview of AutoGCN

Figure 4 shows detailed overview of AutoGCN and Algorithms 1 and 2 shows training and prediction algorithms of AutoGCN. A set of graph $\mathbf{G}$ represents graphs $G_m = (\mathcal{V}, \mathcal{E}, \mathbf{X}_m)$ where $m \in \{1, ..., M\}$ and $\mathcal{V}$ represents EEG nodes, $\mathcal{E}$ represents edges between nodes, $\mathbf{X}_t \in \mathbb{R}^{N \times d}$ where d is the size of EEG slices. In our experiments we used $d = 100$ for EEG slices and edges are same for all graphs so a single trainable adjacency matrix $\mathbf{W}_\mathbf{A}$ represents graph structure of all graphs in $\mathbf{G}$. AutoGCN consists of 3 layers of Batch Graph Convolution layers then their outputs are flattened to a vector then fed into a fully connected layer to compute final results $\mathbf{Y}_t$. Basicly, AutoGCN takes G as inputs and outputs each graph's labels.

During training phase, first, trainable adjacency matrix $\mathbf{W}_\mathbf{A}$ is initialized using method described in Section 4.1. Then AutoGCN takes node attributes $\mathbf{X}_m$ from training dataset $\mathbf{X}$ and computes their encoding using three batch graph convolutional layers. Then new node encodings are fed into a fully connected layer then a Softmax layer to compute their final class probabilities. Finally, a cross-entropy loss function is used to compute the loss of AutoGCN and uses backprogation algorithm to update both parameters of AutoGCN and trainable adjacency matrix $\mathbf{W}_\mathbf{A}$. We used Adam optimizer with learning rate of 0.001 for gradient update and used StepLR learning rate scheduler to reduce multiply learning rate by 0.9 ever 10 epochs. 300 epochs were used to train AutoGCN and parameters with best validation accuracy was chosen as best model and used this model for predicting test dataset. Final AutoGCN model has 3 layer batch graph convolution layers with 512, 1024, 512 neurons each in order to compute 3-hop message passing and aggregation and one fully connected layer with 4 neurons for each classes. Total number of trainable parameters of AutoGCN is 1236996.

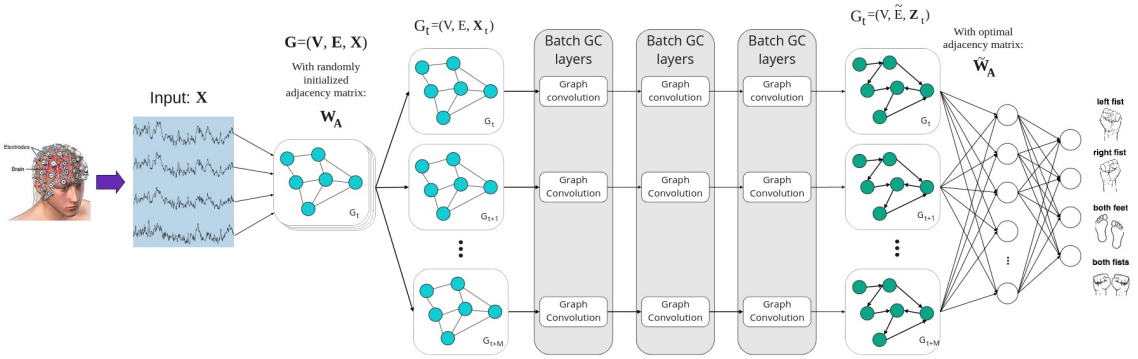


Figure 4: Overview of AutoGCN. First, adjacency matrix $\mathbf{W_A} \in \mathbb{R}^{N \times N}$ of graph $\mathbf{G}$ is randomly initialized then its diagonal values are set to 1. During training phase, batch of M number of graphs are fed into three Batch Convolution layers to train both $\mathbf{W_A}$ and AutoGCN parameters. After training, AutoGCN learns optimal adjacency matrix $\tilde{\mathbf{W}}_\mathbf{A}$ of graph $\mathbf{G}$ and optimal parameters for classifying motor movement imagery of left hand, right hand, both hand and feet movements. Motor motor movement imagery classification accuracy of AutoGCN is also used to evaluate reliability of learned adjacency matrix $\tilde{\mathbf{W}}_\mathbf{A}$.

Algorithm 1: Training algorithm for AutoGCN

Input: Training data $\mathbf{X}$ and target $\mathbf{Y}$
Output: Learned model parameters of AutoGCN and $\tilde{\mathbf{W}}_\mathbf{A}$

Initialize $\mathbf{W_A}$ using Glorot initialization;
Fill diagonal of $\mathbf{W_A}$ to 1;
for $i \in \{1, 2, ..., epochs\}$ **do**
  for $m \in \{1, ..., batches\}$ **do**
    Get batch $\mathbf{X}_m$ and batch $\mathbf{Y}_m$ from $\mathbf{X}$ and $\mathbf{Y}$;
    $\mathbf{Z}_1 \leftarrow \text{BatchGraphConv1}(\mathbf{X}_m, \tilde{\mathbf{W}}_\mathbf{A})$;
    $\mathbf{Z}_2 \leftarrow \text{BatchGraphConv2}(\mathbf{Z}_1, \tilde{\mathbf{W}}_\mathbf{A})$;
    $\mathbf{Z}_3 \leftarrow \text{BatchGraphConv3}(\mathbf{Z}_2, \tilde{\mathbf{W}}_\mathbf{A})$;
    $\hat{\mathbf{Y}}_t \leftarrow \text{FC}(\mathbf{Z}_3)$;
    $loss \leftarrow CrossEntropy(\mathbf{Y}_m, \hat{\mathbf{Y}_m})$;
    Feedback update AutoGCN parameters and $\mathbf{W_A}$;
  end
end

Algorithm 2: Prediction algorithm for AutoGCN

Input: Test data $\mathbf{X}$ and learned adjacency matrix $\tilde{\mathbf{W}}_\mathbf{A}$
Output: Predicted targets $\mathbf{Y}$

for $m \in \{1, ..., batches\}$ **do**
  Get batch $\mathbf{X}_t$ and batch $\mathbf{Y}_t$ from $\mathbf{X}$ and $\mathbf{Y}$;
  $\mathbf{Z}_1 \leftarrow \text{BatchGraphConv1}(\mathbf{X}_m, \tilde{\mathbf{W}}_\mathbf{A})$;
  $\mathbf{Z}_2 \leftarrow \text{BatchGraphConv2}(\mathbf{Z}_1, \tilde{\mathbf{W}}_\mathbf{A})$;
  $\mathbf{Z}_3 \leftarrow \text{BatchGraphConv3}(\mathbf{Z}_2, \tilde{\mathbf{W}}_\mathbf{A})$;
  $\hat{\mathbf{Y}}_t \leftarrow \text{FC}(\mathbf{Z}_3)$;
end
return $\hat{\mathbf{Y}}$

Table 1: Overview of Physionet Motor Movement Imagery dataset

Physionet Motor Movement Imagery dataset	
Number of subjects	105
Number of runs	16
number of channels	64
Sampling frequency	160
Baseline tasks	Eyes open, eyes closed
Motor execution tasks	left hand, right hand, both hands, both feet
Motor imagery tasks	left hand, right hand, both hands, both feet

5 Experiments

5.1 Physionet Motor Movement Imagery Dataset

We used PhysioNet Motor Movement Imagery Dataset to evaluate AutoGCN. Physionet Motor Movement Imagery dataset is a dataset consisting of over 1500 one and two miunte EEG recordings obtained from 109 volunteers recorded using PhysioNet's proprietary P300 EEG recording device. However, subjects 88, 89, 92, 100 were excluded due to bad recordings. Each subject performed 14 experimental runs: two one-minute baseline runs (one with eyes open, one with eyes closed), and three two-minute runs of each of the four following tasks:

1. A target appears on either the left or the right side of the screen. The subject opens and closes the corresponding fist until the target disappears. Then the subject relaxes.

2. A target appears on either the left or the right side of the screen. The subject imagines opening and closing the corresponding fist until the target disappears. Then the subject relaxes.

3. A target appears on either the top or the bottom of the screen. The subject opens and closes either both fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears. Then the subject relaxes.

4. A target appears on either the top or the bottom of the screen. The subject imagines opening and closing either both fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears. Then the subject relaxes.

The data were provided in EDF+ format containing EEG signals, each samples at 160 samples per second, and an annotation channel. The EEGs were recorded from 64 electrodes as per the international 10-10 system (excluding electrodes NZ, F9, F10, FT9, FT10, A1, A2, TP9, TP10, P9, and P10). Overview of the PhysioNet dataset is shown in Table 1. In order to find how number of subjects affect trained adjacency matrix we created 3 different datasets with 105 subjects, 50 subjects, 20 subjects and 80% of each dataset were used for training 10% for validation and 10% for testing.

5.2 Dataset preprocessing

Because we want to classify motor movement imagery we used recording with motor imagery tasks. Let $\mathbf{X} \in \mathbb{R}^{M \times N}$ be motor imagery data of Physionet dataset and $\mathbf{Y} \in \mathbb{R}^{M}$ be their class labels. In order to capture its temporal dependency we sliced $\mathbf{X}$ and $\mathbf{Y}$ into time-series sets with length d. Then final dimensions of dataset and labels used in

Table 2: Comparison models

Model name	Architecture	Trainable parameters
FCN	FC→FC→FC	1232900
CNN	Conv→FC	1264684
RNN	LSTM→LSTM	893956
GCN	BatchGC→BatchGC→BatchGC	1232900
GCRAM	Conv→Maxpool→LSTM→LSTM-Attention	1462316
AutoGCN	BatchGC→BatchGC→BatchGC	1236996
AutoGCRAM	BatchGC→Conv→Maxpool→LSTM→LSTM-Attention	795948

AutoGCN becomes $\mathbf{X} \in \mathbb{R}^{\frac{M}{d} \times N \times d}$ and $\mathbf{Y} \in \mathbb{R}^{\frac{M}{d}}$ respectively. We used each label slice's last label value as a label for slice $\mathbf{X}_t$.

Also, we used standard scaling method to scale dataset $\mathbf{X}$ on its mean and standard deviation in order to help with AutoGCN's generalization and improve training speed.

5.3 Comparison models

In order to compare the results of AutoGCN we implemented 3-layer FCN wiith 512, 1024 and 512 neurons per layer, CNN with single 40 kernel convolutional layer with 64×45 kernel sizes and RNN with 2-layer LSTM with 64 cells in each unit as baseline models. All baseline models are modified to work with time-series data and has similar amount of parameters to AutoGCN. In order to compare effectiveness of AutoGCN's learned adjacency matrices, we also implemented GCN and G-CRAM with N-graph method proposed by Dalin Zhang et al. Also, we implemented our version of G-CRAM with same learnable adjacency matrix and same initialization method as AutoGCN and 1-layer batch graph convolution layer called AutoGCRAM to find if our method can be used to improve other DL based models. Architecture and number of trainable parameters of comparison models are shown in Table 2.

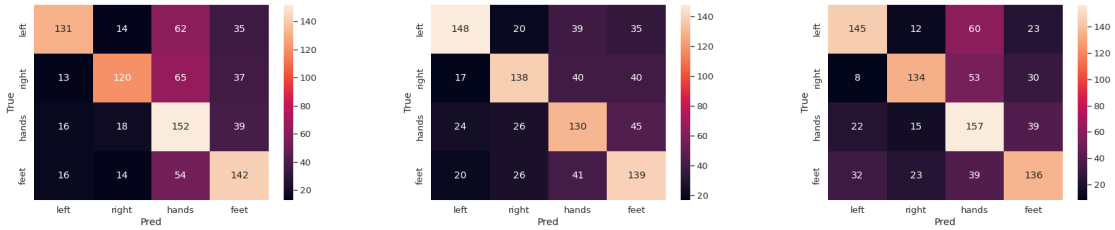
5.4 Results

Table 3 highlights the experimental results of comparison models and our model and their average accuracy and standard deviations on the test dataset with total of 10 runs for each model. And Figure 5 shows confusion matrices of top 3 models: AutoGCN, G-CRAM and AutoGCRAM models. Firstly, we can see from confusion matrices in Figure 5 that AutoGCN and AutoGCRAM have decent performance in classifying 4-class motor movement imagery EEG data. Furthermore, despite AutoGCN not achieving best results AutoGCRAM which uses trainable adjacency matrix and 1-layer batch convolution layer method proposed in our work achieved highest mean accuracy in 105 subject, 20 subjects datasets and second best accuracy in in 50 subject datasets. Furthermore, AutoGCN model achieved significantly higher results than regular GCN with N-graph structure. These results suggest that N-graph method proposed by Dalin Zhang et al. is in fact not optimal and learned adjacency matrix from our proposed models generate optimal adjacency matrices.

Results in Table 3 also suggest that EEG dataset recorded from multiple different subjects do not necessarily hurt performance of AutoGCN and AutoGCRAM models. Initially, we speculated that each subjects' brain region that activate during motor movement imagery task could differ from subject to subject depending on their age, gender, ethnicity, nationality etc. However, the fact that accuracy of AutoGCN on 105 subjects is higher than 50 subjects suggest that most people, if not all, have same regions that

Table 3: Accuracy results on PhysioNet motor movement imagery dataset. Bold, underlined and italic numbers show first, second and third best models respectively.

Model name	Accuracy(105 subjects)	Accuracy(50 subjects)	Accuracy(20 subjects)
FCN	50.2 ± 1.37	50.0 ± 2.61	49.8 ± 3.52
CNN	54.2 ± 1.51	54.3 ± 2.55	51.4 ± 2.87
RNN	40.4 ± 3.0	37.0 ± 3.33	34.8 ± 4.03
GCN	48.1 ± 1.63	49.0 ± 1.96	46.2 ± 5.16
GCRAM	$\underline{56.9} \pm \underline{2.27}$	$\mathbf{56.8 \pm 1.81}$	$\underline{54.4} \pm \underline{5.18}$
AutoGCN	$\mathit{56.0 \pm 1.59}$	$\mathit{55.5 \pm 2.44}$	$\mathit{54.0 \pm 2.75}$
AutoGCRAM	$\mathbf{58.8 \pm 1.75}$	$\underline{56.7} \pm \underline{2.84}$	$\mathbf{56.1 \pm 5.27}$



(a) AutoGCN confusion matrix

(b) G-CRAM confusion matrix

(c) AutoGCRAM confusion matrix

Figure 5: Confusion matrices of G-CRAM, AutoGCN and AutoGCRAM on test set.

activate during EEG MMI tasks. Another explanation could be that motor movement imagery task is too primitive task to compare brain regions' activations. Training AutoGCN on more complex BCI tasks then comparing their results could be used to find if effective connectivity of the brain differ from one subject to another.

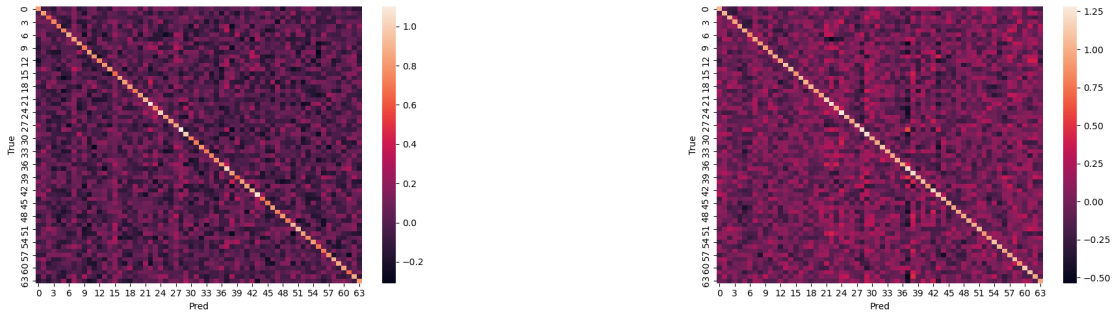
5.5 Trained adjacency matrix and effective connectivity

Figure 6 shows heatmap of learned adjacency matrices $\tilde{\mathbf{W}}_{\mathbf{A}}$ of AutoGCN model and AutoGCRAM model. By look at the heatmaps we can see that there are several lighter and darker columns in each heatmap. This could indicate importance of these EEG nodes on right hand, left hand, both hands and both feet imagery tasks. Because values of $\tilde{\mathbf{W}}_{\mathbf{A}}$ are non-binary graph constructed using $\tilde{\mathbf{W}}_{\mathbf{A}}$ would become fully connected directed weight multigraph. In order to visualize important edges we used simple thresholding to change values edges with less influence to 0 using following formula:

$$A_{ij} = \begin{cases} A_{ij} & \text{if } |A_{ij}| < T \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where T is threshold value and is manually defined. By constructing graph structure of EEG nodes using (8) we can visualize EEG nodes' effective connectivity or how much they influence each other. Figure 7 show directed graphs constructed by thresholding $\tilde{\mathbf{W}}_{\mathbf{A}}$. Blue and red edges represent positive and negative edge values respectively .

By looking at Figure 7a and Figure 7c we can see that nodes are connected to each other regardless of physical distance as we suggested in Section 3.4. When subject is executing a specific task, electrical activity is found all over the brain and its not constrained to in regions of the brain that are specific to the task. Also, by looking at Figure 7b and Figure 7d we can see that nodes that upper right and lower right electrodes affect other nodes heavily. This suggests that focusing BCI on these regions could prove to be more



(a) AutoGCN adjacency matrix

(b) AutoGCRAM adjacency matrix

Figure 6: Heatmap showing trained adjacency matrices of AutoGCN and AutoGCRAM.

productive than using subject's whole scalp. Also, in Figure 7b we can also see that many nodes have negative effect on node F8 and positive effect on node FT7.

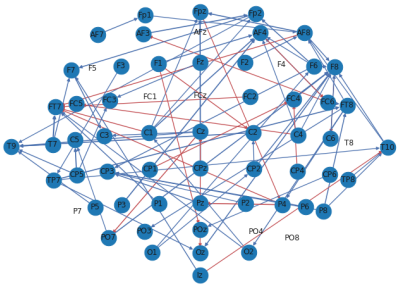
As we can see, supervised effective connectivity prediction using models such as AutoGCN and AutoGCRAM helps tremendously with gaining knowledge about effective connectivity of the human brain. In turn, these knowledge can be used design even better and efficient BCI.

5.6 Results on MNIST data

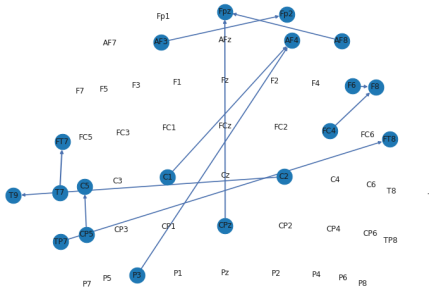
Even though AutoGCN was designed to work with time-series EEG data it can also work with other data structures such as image data. In this experiment we used MNIST data to train AutoGCN and visualized its learned adjacency matrix.

AutoGCN works with input $\mathbf{X}$ of dimensions $R^{M\times N\times d}$ data. Another data with similar dimensions is MNIST data where it is a set of images $\mathbf{X}_{MNIST}$ of size $\mathbb{R}^{M\times 28\times 28}$. We can convert single MNIST image into graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ where by setting $\mathcal{V}$ as row indices and $\mathbf{X} \in \mathbb{R}^{28\times 28}$ as a node features with row pixel values. Then we train AutoGCN using graph converted MNIST images. Figure 8 shows AutoGCN's learned adjacency matrix on MNIST dataset and graphs are constructed by setting $T = 0.08$ and $T = 0.1$. Test accuracy of AutoGCN on MNIST was over 97%.

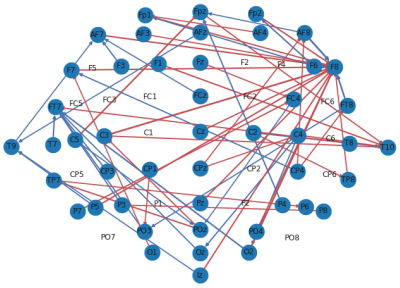
By looking at the learned adjacency matrix in Figure 8a we can see that AutoGCN has larger weights on rows/nodes 4 to 25. This makes sense because digits in MNIST images are usually drawn with small whitespace on top and bottom and digits are usually drawn between rows 4 to 25. Heavy weight values around these areas suggest that that AutoGCN can also be used to visualize effective connectivity of images.



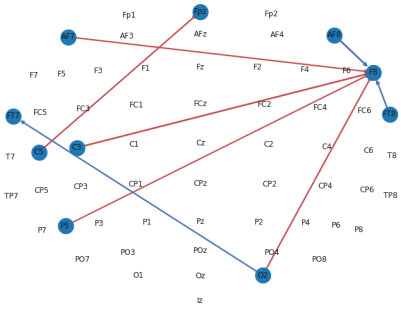
(a) AutoGCN graph structure $T > 0.25$



(b) AutoGCN graph structure $T > 0.33$

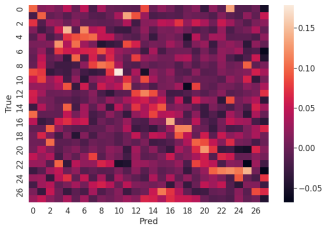


(c) AutoGCRAM graph structure $T > 0.35$

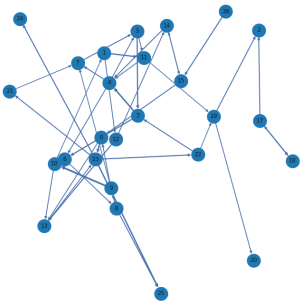


(d) AutoGCRAM graph structure $T > 0.45$

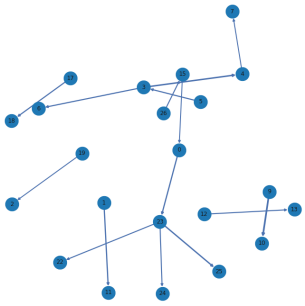
Figure 7: Effective connectivity of EEG nodes constructed using learned adjacency matrix from AutoGCN and AutoGCRAM. Blue arrows and red arrows mean positive and negative edge values respectively.



(a) AutoGCN Learned adjacency matrix on MNIST dataset



(b) AutoGCN graph structure $T > 0.08$



(c) AutoGCN graph structure $T > 0.1$

Figure 8: Effective connectivity of MNIST constructed using learned adjacency matrix from AutoGCN. In this case, each node i represents i-th row of a MNIST image.

6 Conclusion

This study surveys recent works in BCI methods and outlined their problems:

1. Both traditional and deep learning based methods does not take into effective connectivity of EEG nodes.

2. Some works that do take effective connectivity manually define oversimplified connectivities.

In order to solve these problems we proposed AutoGCN model which uses trainable adjacency matrix and to represent effective connectivity of EEG nodes and use batch graph convolution layers to compute encodings of temporal EEG graphs. Results on Physionet EEG dataset show that learned adjacency matrix of AutoGCN is more optimal than manually defined adjacency matrix from previous work. Also, a previous work modified by AutoGCN called AutoGCRAM showed higher accuracy than all other comparison models. These results show that learned adjacency matrices from AutoGCN and AutoGCRAM are optimal than previous works. Furthermore, visualization of adjacency matrices trained on Physionet EEG dataset could be used to further analyze effective connectivity of EEG nodes. Knowledge gained from this process could be used to further improve BCI methods.

We also show that AutoGCN is not only limited EEG motor movement imagery classification task by training AutoGCN on MNIST dataset. Visualization of trained adjacency matrices show that they in fact learn effective connectivity of nodes.

7 Acknowledgment

I would like to thank my supervisor, Prof. Tsuyoshi Murata for providing valuable comments, feedbacks and providing us with GPU machines for this research project.

I would like to thank my everyone in Murata laboratory including lab mates and secretaries and supervisors for supporting me. I would like to thank my lab mate Nguyen Thai Hoang for providing me with especially useful advice on my research.

I wish to thank my friends, family who have supported me, encouraged me, and motivated me. I appreciate everything they have done for me.

References

[1] Kai Keng Ang, Zhang Yang Chin, Haihong Zhang, and Cuntai Guan. Filter bank common spatial pattern (FBCSP) in brain-computer interface. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, June 2008.

[2] A. Biasiucci, R. Leeb, I. Iturrate, S. Perdikis, A. Al-Khodairy, T. Corbet, A. Schnider, T. Schmidlin, H. Zhang, M. Bassolino, D. Viceic, P. Vuadens, A. G. Guggisberg, and J. d. R. Millán. Brain-actuated functional electrical stimulation elicits lasting arm motor recovery after stroke. *Nature Communications*, 9(1), June 2018.

[3] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

[4] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[5] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[7] Nancy Kanwisher. The human brain: Class 3 neuroanatomy. *MIT OpenCourseWare*, Spring 2019.

[8] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016.

[9] Zoltan J. Koles, Michael S. Lazar, and Steven Z. Zhou. Spatial patterns underlying population differences in the background EEG. *Brain Topography*, 2(4):275–284, 1990.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[11] Ravikiran Mane, Tushar Chouhan, and Cuntai Guan. BCI for stroke rehabilitation: motor and beyond. *Journal of Neural Engineering*, 17(4):041001, aug 2020.

[12] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[13] G. Schalk, D.J. McFarland, T. Hinterberger, N. Birbaumer, and J.R. Wolpaw. Bci2000: a general-purpose brain-computer interface (bci) system. *IEEE Transactions on Biomedical Engineering*, 51(6):1034–1043, 2004.

[14] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank

Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for EEG decoding and visualization. *Human Brain Mapping*, 38(11):5391–5420, August 2017.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[16] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[17] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020.

[18] Dalin Zhang, Kaixuan Chen, Debao Jian, and Lina Yao. Motor imagery classification via temporal attention cues of graph embedded eeg signals. *IEEE Journal of Biomedical and Health Informatics*, 24(9):2570–2579, 2020.

[19] Dalin Zhang, Lina Yao, Kaixuan Chen, and Jessica Monaghan. A convolutional recurrent attention model for subject-independent eeg signal analysis. *IEEE Signal Processing Letters*, 26(5):715–719, 2019.

[20] Dalin Zhang, Lina Yao, Kaixuan Chen, and Sen Wang. Ready for use. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, October 2018.

[21] Dalin Zhang, Lina Yao, Kaixuan Chen, Sen Wang, Pari Delir Haghighi, and Caley Sullivan. A graph-based hierarchical attention model for movement intention detection from eeg signals. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(11):2247–2253, 2019.