2022 44th Annual International Conference of
the IEEE Engineering in Medicine & Biology Society (EMBC)
Scottish Event Campus, Glasgow, UK, July 11-15, 2022

2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC) | 978-1-7281-2782-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/EMBC48229.2022.9871984

EEG-GAT: Graph Attention Networks for Classification of Electroencephalogram (EEG) Signals

Andac Demir, Toshiaki Koike-Akino, Ye Wang, and Deniz Erdoğmuş

Abstract— **Graph neural networks (GNN) are an emerging framework in the deep learning community. In most GNN applications, the graph topology of data samples is provided in the dataset. Specifically, the graph shift operator (GSO), which could be adjacency, graph Laplacian, or their normalizations, is known a priori. However we often have no knowledge of the grand-truth graph topology underlying real-world datasets. One example of this is to extract subject-invariant features from physiological electroencephalogram (EEG) to predict a cognitive task. Previous methods use electrode sites to represent a node in the graph and connect them in various ways to hand-engineer a GSO e.g., i) each pair of electrode sites is connected to form a complete graph, ii) a specific number of electrode sites are connected to form a k-nearest neighbor graph, iii) each pair of electrode site is connected only if the Euclidean distance is within a heuristic threshold. In this paper, we overcome this limitation by parameterizing the GSO using a multi-head attention mechanism to explore the functional neural connectivity subject to a cognitive task between different electrode sites, and simultaneously learn the unsupervised graph topology in conjunction with the parameters of graph convolutional kernels.**

Index Terms— **Graph neural networks (GNN), Graph shift operators (GSO), Convolutional neural networks (CNN), electroencephalogram (EEG) classification.**

I. Introduction

Convolutional Neural Networks (CNN) have been widely used to solve several problems with high performance, including, but not limited to, image classification, object detection, semantic segmentation, data reconstruction, and super-resolution. For most of these tasks, data samples lie in a regular domain and have the representation of a grid-like structure such as a 2-dimensional pixel format. However, the preponderance of machine learning applications require learning from graphs with irregular structures. Graph Neural Networks (GNN) have been applied to many such machine learning tasks ranging from simulating complex physical systems [1], predicting molecular interactions [2], modeling social networks [3], and exploring causal inference in a probabilistic model defined over a directed acyclic graph (DAG) [4]. Although the ultimate goal of GNNs is to generalize the concept of convolution into graphs, there are numerous variants of GNNs with modifications on the convolution operator and hierarchical pooling mechanism to better propagate the local neighborhood information or the use of skip connections to alleviate the problem of vanishing gradients in the earlier layers of deep networks.

Two major methods for GNNs are spectral networks and spatial networks. Spectral methods use graph signal processing theory by applying the convolution operator in the spectral domain [5], [6]. First, the graph structured data $\boldsymbol{x}$ is converted to the spectral domain via the graph Fourier transform $\mathcal{F}$ by computing the eigendecomposition of the normalized graph Laplacian $\boldsymbol{L}$ (evaluated using the degree matrix $\boldsymbol{D}$ and adjacency matrix $\boldsymbol{A}$). Then, convolution in the time domain is conducted by multiplication in the spectral domain. The output is converted to the time domain via inverse the Fourier transform $\mathcal{F}^{-1}$ at the final stage, e.g., as follows:

$$\begin{aligned}&\boldsymbol{L}=\boldsymbol{I}-\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}=\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T,\\ &\mathcal{F}(\boldsymbol{x})=\boldsymbol{U}^T\boldsymbol{x}, \qquad \mathcal{F}^{-1}(\boldsymbol{x})=\boldsymbol{U}\boldsymbol{x},\\ &\boldsymbol{g}*\boldsymbol{x}=\mathcal{F}^{-1}(\mathcal{F}(\boldsymbol{g})\cdot\mathcal{F}(\boldsymbol{x}))=\boldsymbol{U}(\boldsymbol{U}^T\boldsymbol{g}\cdot\boldsymbol{U}^T\boldsymbol{x}).\end{aligned} \tag{1}$$

Since spectral GNNs depend on the eigendecomposition of the graph Laplacian, which is evaluated from the graph structure, a spectral GNN cannot be trained on a dataset where the graph topology of data samples is unsupervised. Other primary problems with spectral GNNs include that they omit the edge features, and increase the computational overhead, in particular for large graphs with an abundant number of nodes.

One way to solve this problem is to use spatial GNNs. Spatial GNNs apply convolutions directly on the graph by projecting nodes' feature vector onto a new subspace and aggregating K-hop node representations using a permutation-invariant and injective function: `AGGREGATE`. At the final stage of spatial graph convolution, a representation vector of the entire graph is learned via another permutation-invariant and injective function: `READOUT` [7]–[9]. A spatial GNN propagates the information from K-hop neighborhood embeddings, $\boldsymbol{h}_u^{k-1}, \forall u \in N(v)$, and learns an aggregated representation, $\boldsymbol{h}_{N(v)}^{k}$, where $N(v)$ denotes the set of node v's K-hop neighbors. The current node representation vector denoted as $\boldsymbol{h}_v^{k-1}$ is concatenated with the aggregated representation and then the outcome is passed through a dense layer, $\mathbf{W}^k$, followed by a nonlinear activation, σ. Algorithm 1 summarizes the procedure [9].

CNN architectures have been frequently used to extract the subject and session invariant features from EEG signals to perform classification tasks [13]. There are two principal methodologies to train CNN architectures on EEG signals:

1) Each EEG trial is passed to a CNN model as a pseudo grayscale image, $\mathbb{R}^{C\times T}$, where C denotes number of

A. Demir and D. Erdoğmuş are with Cognitive Systems Laboratory, Electrical and Computer Engineering Department, Northeastern University, Boston, MA 02115, USA (e-mail: {demir.a, erdogmus}@ece.neu.edu).

T. Koike-Akino and Y. Wang are with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA (e-mail: {koike, yewang}@merl.com).

A. Demir conducted this research when he was an intern at MERL.

TABLE I: Different choices of GSO to propagate neighborhood information in graphs [10]

$S = \{m_1, m_2, m_3, e_1, e_2, e_3, a\}$	$\gamma(\boldsymbol{A}, S)$	**Choice of GSO**
$\{0, 1, 0, 0, 0, 0, 0\}$	$\boldsymbol{A}$	Adjacency Matrix
$\{1, -1, 0, 1, 0, 0, 0\}$	$\boldsymbol{D} - \boldsymbol{A}$	Unnormalized Laplacian Matrix
$\{1, 1, 0, 1, 0, 0, 0\}$	$\boldsymbol{D} + \boldsymbol{A}$	Signless Laplacian Matrix [11]
$\{0, 1, 0, 0, -1, 0, 0\}$	$\boldsymbol{D}^{-1}\boldsymbol{A}$	Mean Aggregation Operator [8]
$\{0, -1, 1, 0, -1, 0, 0\}$	$\boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A}$	Random Walk Normalized Laplacian Matrix
$\{0, 1, 0, 0, -\frac{1}{2}, -\frac{1}{2}, 1\}$	$\boldsymbol{D}_1^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}_1^{-\frac{1}{2}}$	Normalized Adjacency Matrix [12]
$\{0, -1, 1, 0, -\frac{1}{2}, -\frac{1}{2}, 0\}$	$\boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$	Symmetric Normalized Laplacian Matrix

Algorithm 1 Learning graph embedding via spatial GNN

```
$\boldsymbol{h}_v^0 \leftarrow \mathbf{X}_v, \forall v \in V$  Initialize a representation vector for each node
for $k = 1 \ldots K$ do
    for $v \in V$ do
        $\boldsymbol{h}_{N(v)}^k \leftarrow \texttt{AGGREGATE}_k(\{\boldsymbol{h}_u^{k-1},\ \forall u \in N(v)\})$
        $\boldsymbol{h}_v^k \leftarrow \sigma\big(\mathbf{W}^k \cdot \texttt{CONCAT}(\boldsymbol{h}_v^{k-1}, \boldsymbol{h}_{N(v)}^k)\big)$
    $\boldsymbol{h}_v^k \leftarrow \boldsymbol{h}_v^k / \|\boldsymbol{h}_v^k\|_2, \forall v \in V$
$\boldsymbol{h}_G \leftarrow \texttt{READOUT}(\{\boldsymbol{h}_v^K,\ \forall v \in V\})$
```

EEG channels, and T denotes number of discretized time samples, and 2D convolution kernels are slid over the 2D input data.

2) Each EEG trial is passed to a CNN model as a multi-channel pseudo image, $\mathbb{R}^{C \times 1 \times T}$, and 1D convolutions are applied along the time axis of each EEG trial.

Although CNNs have shown success in the classification of several physiological datasets, they represent relationships between EEG channels with the spatial or channel dimensions of an image, both of which oversimplify the physiological structure and ignores the graph topology that may be inferred from observations. Spatial GNNs are one way to overcome this problem by representing EEG headsets as an undirected graph. Let $G = (V, E, W)$ represent a triplet of vertices, edges, and weights of a graph, where V denotes the set of vertices, E denotes the set of edges, W denotes the set of weights, and $|V| = C$. In our previous work [14], we assume an electroencephalogram (EEG) trial, collected from C electrodes and T discretized time samples, can be represented as graph data $\boldsymbol{X} \in \mathbb{R}^{C \times T}$, where the nth row of $\boldsymbol{X}$ presents the T-dimensional feature vector of node n. For the GNN, graph edges between electrode sites can represent the functional neural connectivity between different sites and lobes of the brain for a specific cognitive activity. Several spatial GNN networks, e.g., GraphSage, Graph Isomorphism Network (GIN), SortPool, EdgePool, SagPool and Set2Set have been previously modified and tested on different EEG datasets [14]. Nevertheless, all of these models require an *a priori* knowledge of the graph shift operator (GSO), and the GSO is fixed for a diverse cohort of participants and different data acquisition sessions. Typically, they formulate an adjacency matrix under some assumptions: i) each pair of electrode sites is connected (complete graph), ii) a k-nearest neighbor graph (NNG), where each electrode site is connected to the k closest other electrode sites, iii) each pair of electrode sites is connected, if the Euclidean distance between is lower than a heuristic threshold, and iv) self-loops are included for conditions i)-iii).

The graph topology of observations is an invaluable source of information for training machine learning models on graphs. Our research focuses on parameterizing the edge weights via a multi-head attention mechanism instead of using a heuristically structured and fixed GSO, hence we can assign a different importance to every edge in a complete graph. When the edge weights are non-uniform, we can set a different importance to every edge while aggregating neighboring node representations to fuse local and global information. Parameterizing the edge weights and learning an optimal GSO as a topology representation lead to the exploration of neural connectivity factors between different electrode sites peculiar to a specific cognitive task as well as significant gain in classification performance.

II. Related Work

Previous studies listed in Table I investigate the selection of GSO by choosing a different set of parameter values. Although the specific choice of GSO matters in theory, there is no significant difference in the reported experimental results.

Definition 1. *According to [16], a GSO, denoted by the matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$, where n is the number of nodes, satisfies $\boldsymbol{S}_{vu} = 0$ if $v \neq u$ and $(v, u) \notin E$.*

Definition 2. *A parameterized GSO, denoted by $\gamma(\boldsymbol{A}, S)$, is defined as follows [10],*

$$\begin{aligned} \gamma(\boldsymbol{A}, S) &= m_1 \boldsymbol{D}_a^{e_1} + m_2 \boldsymbol{D}_a^{e_2} A_a D_a^{e_3} + m_3 \boldsymbol{I}, \\ \boldsymbol{A}_a &= \boldsymbol{A} + a\boldsymbol{I}, \quad \boldsymbol{D}_a = \operatorname{diag}(\boldsymbol{A}_a \boldsymbol{1}), \end{aligned} \tag{2}$$

where $S = \{m_1, m_2, m_3, e_1, e_2, e_3, a\}$ is a parameter set.

The set S consists of scalar multiplicative parameters $m_{1:3}$, scalar exponential parameters $e_{1:3}$ and a that is a non-negative integer to determine the number of self loops in the graph. Since exponential parameters $e_{1:3}$ are only applied to diagonal matrices, a parameterized GSO can be efficiently computed and optimized. Experimental analysis in [10] shows the optimization of GSO parameters is numerically stable over epochs, however the learned values of these parameters is very vulnerable to the different initializations of S. Specifically, if the parameterized GSO is initialized

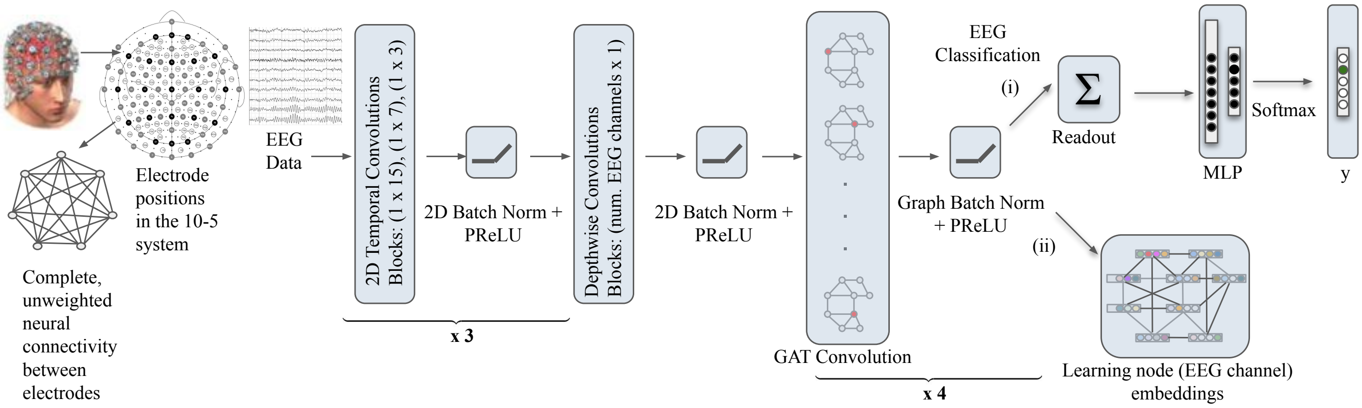
Electrode positions in the 10-5 system
Complete, unweighted neural connectivity between electrodes
EEG Data
2D Temporal Convolutions Blocks: (1 x 15), (1 x 7), (1 x 3)
2D Batch Norm + PReLU
Depthwise Convolutions Blocks: (num. EEG channels x 1)
x 3
2D Batch Norm + PReLU
GAT Convolution
x 4
Graph Batch Norm + PReLU
EEG Classification
(i)
Σ
Readout
MLP
Softmax
y
(ii)
Learning node (EEG channel) embeddings

Fig. 1: **EEG-GAT framework** scales up EEGNet [13] with the addition of graph attentional layers. EEG-GAT starts with 2D temporal convolutional operations to learn frequency filters and explore short, medium and long term temporal dependencies respectively, convolving EEG trials with kernels of size (1×15), (1×7), (1×3). To preserve the spatial dimensions of the input EEG trial, zero padding of size (0×7), (0×3), (0×1) is added. Number of convolutional kernels: $30, 60, 90$ increases in each layer. This is followed by depthwise separable convolutional operations to learn frequency specific spatial filters with kernels of size $C \times 1$. In order to preserve the spatial dimensions of the EEG trial, zero padding of size $(\text{floor}(C/2), 0)$ is added. Depthwise separable convolution is frequently used in mobile vision applications, since they have fewer parameters which reduces the computational footprint while improving the model throughput and mitigating the risk of overfitting. The objective of using depthwise separable convolutions in EEG-GAT is to capture the frequency specific spatial information of the EEG data in an efficient way. After each convolutional layer, we apply batch normalization along the feature map dimension to improve gradient flow through the network and reduce the risk of overfitting. Then, we apply PReLU activation to introduce non-linearity. GAT convolution [15] is operated over a complete, undirected graph, where nodes are the EEG electrodes according to the international 10-20, 10-10 or 10-5 systems for recording and node features are the spatio-temporal features extracted by the CNN backbone. Node representations are computed by aggregating information from a node's K-hop neighbors. At the final stage of EEG-GAT, there are 2 possible outcomes: Following (i), we can apply a global mean pooling operator, which averages the node representations in order to return a batch-wise graph level output. Then a multi-layer perceptron (MLP) can classify the graph representation vector. Otherwise, following (ii), we can use node representation vectors to perform node classification, which is useful for EEG channel selection/sorting to reduce computational overhead while analyzing EEG data.

as an adjacency matrix, the optimal values learned for GSO parameters after convergence would be quite distinct from the learned GSO parameters if GSO is initialized as a random walk Laplacian. Additionally, there is no constraint on the optimization objective to satisfy the condition $S_{vu} = 0$ if $v \neq u$ and $(v, u) \notin E$.

Example 1. *A special form of spatial GNN method, GIN, updates node embeddings aggregating information from K-hop neighbors based on the following propagation rule,*

$$\boldsymbol{h}_v^k = \sigma\Bigg(\boldsymbol{h}_v^{k-1}\mathbf{W}^k + \sum_{\forall u \in N(v)} \boldsymbol{h}_u^{k-1}\mathbf{W}^k\Bigg). \qquad (3)$$

Using the Definition 2, we can modify (3) to formulate a propagation rule for GINs that use a parameterized GSO:

$$\begin{aligned} \boldsymbol{h}_v^k &= \sigma\Bigg(\alpha_v \boldsymbol{h}_v^{k-1}\mathbf{W}^k + m_2 \sum_{\forall u \in N(v)} e_{vu}\boldsymbol{h}_u^{k-1}\mathbf{W}^k\Bigg), \\ \alpha_v &= m_1(\boldsymbol{D}_a)_v^{e_1} + m_3, \quad e_{vu} = (\boldsymbol{D}_a)_v^{e_2}(\boldsymbol{D}_a)_u^{e_3}. \end{aligned} \qquad (4)$$

While α_v represents the node importance factor, e_{vu} represents the edge weight between nodes v and u.

III. EEG-GAT Framework

Our framework EEG-GAT, as illustrated in Fig. 1, uses the EEGNet architecture as a backbone network to extract frequency specific spatial features from EEG trials and the graph attentional operators to explore the strength of cognitive activity peculiar to an EEG classification task between each pair of electrode site. The latter allows us to capture the intricate interactions in the brain network projected onto a map of scalp electrodes according to the electrode locations of international 10-20, 10-10 and 10-5 systems for EEG recording.

Graph attention networks (GATs) offer an alternative approach to learning a parameterized GSO, since the main objective of GATs is to implicitly evaluate the importance of node v's features to node u by exploring attention weights α_{uv}. We observed that parameters of GAT convolutional layers initialized by Xavier uniform [17] converged model parameters are robust against arbitrary model initializations unlike the original parameterized GSO study.

Algorithm 2 summarizes the training procedure of a single GAT layer. The output of CNN backbone in EEG-GAT preserves the spatial size of an EEG trial. The input to a GAT layer is the set of node representation vectors $\mathbf{X}_v$, $\forall v \in V$.

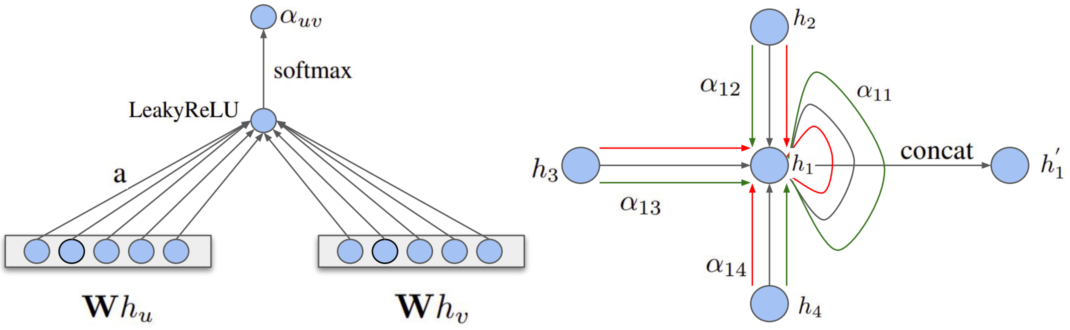

Fig. 2: Left figure illustrates the linear transformation, parameterized by matrix $\mathbf{W}$ applied to 2 node feature vectors. The outputs are concatenated, passed through a single layer feedforward network a and normalized by softmax to compute the attention coefficients, α_{vu}. Right figure illustrates the GAT layer that employs multi-head attention mechanism. There are 3 independent attention coefficients which affect the importance nodes $\boldsymbol{h}_2$, $\boldsymbol{h}_3$ and $\boldsymbol{h}_4$ to the node $\boldsymbol{h}_1$. Features from each of the neighboring nodes are modulated by these attention coefficients and then concatenated to obtain a new representation for $\boldsymbol{h}_1$: $\boldsymbol{h}_1^{'}$.

To compute the attention coefficients e_{vu}, we apply the same linear transformation, $\mathbf{W}$, on all the node representations. Then, for each neighbor nodes v and u, we concatenate the transformed node representations: $[\boldsymbol{h}_v \| \boldsymbol{h}_u]$, where $\|$ denotes concatenation. This concatenated vector is passed through a single layer feedforward network, $\mathbf{a}$, and the output is passed through LeakyReLU activation. We normalize the attention coefficients using softmax, hence they are comparable. Once normalized, we use attention coefficients, α_{vu} to weigh node representation vectors and aggregate information from neighboring nodes. The output is passed through log-softmax activation, denoted by σ, and model parameters are optimized by minimizing the negative log-likelihood loss. It is illustrated in Fig. 2.

Algorithm 2 Single-head graph attentional (GAT) layer

```
Initialize a representation vector for each node:
$\boldsymbol{h}_v \leftarrow \mathbf{X}_v, \forall v \in V$, $\mathbf{X} \in \mathbb{R}^{|V| \times T}$ where $|V|$ is the number of electrodes and $T$ is equivalent to the number of discretized time samples in an EEG trial and $\mathbf{X}_v \in \mathbb{R}^{T \times 1}$.
$\boldsymbol{h}_v \leftarrow \mathbf{W}\mathbf{X}_v, \forall v \in V$ and $\mathbf{W} \in \mathbb{R}^{T^{'} \times T}$
for $v \in V$ do
    for $u \in N(v)$ do
        $e_{vu} = \mathrm{LeakyReLU}(\mathbf{a}[\boldsymbol{h}_v \| \boldsymbol{h}_u])$, $\mathbf{a} \in \mathbb{R}^{2T^{'}}$
    $\alpha_{vu} = \exp(e_{vu}) / \sum_{j \in N(v)} \exp(e_{vj})$, $\forall u \in N(v)$
$\boldsymbol{h}_v = \sigma(\sum_{u \in N(v)} \alpha_{vu} \boldsymbol{h}_u)$
$\boldsymbol{h}_G \leftarrow \mathtt{READOUT}(\{\boldsymbol{h}_v, \ \forall v \in V\})$
```

Instead of performing a single-head attention, multi-head attention mechanism, first proposed in the implementation of transformer networks, allows the model to combine information from different representation subspaces [18]. Algorithm 3 outlines the utilization of multi-head attention mechanism while training a single GAT layer. The major difference is an intermediate GAT layer that employs multi-head attention mechanism employs K independent single layer feedforward networks, $\mathbf{a}^k$, and concatenates node representations. If the GAT layer is at the end of the network, then node representations modulated by the attention coefficients are averaged.

Algorithm 3 Multi-head graph attentional (GAT) layer

```
Initialize a representation vector for each node:
$\boldsymbol{h}_v^0 \leftarrow \mathbf{X}_v, \forall v \in V$, $\mathbf{X} \in \mathbb{R}^{|V| \times T}$ and $\mathbf{X}_v \in \mathbb{R}^{T \times 1}$.
for $k = 1 \dots K$ do
    $\boldsymbol{h}_v^k \leftarrow \mathbf{W}^k \mathbf{X}_v, \forall v \in V$
    for $v \in V$ do
        for $u \in N(v)$ do
            $e_{vu}^k = \mathrm{LeakyReLU}(\mathbf{a}^k[\boldsymbol{h}_v^k \| \boldsymbol{h}_u^k])$
            $\alpha_{vu}^k = \exp(e_{vu}^k) / \sum_{j \in N(v)} \exp(e_{vj}^k)$, $\forall u \in N(v)$
if GATConv is the final (prediction) layer of the network then
    $\boldsymbol{h}_v^K = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{u \in N(v)} \alpha_{vu}^k \boldsymbol{h}_u^k)$
else
    $\boldsymbol{h}_v^K = \|_{k=1}^{K} \sigma(\sum_{u \in N(v)} \alpha_{vu}^k \boldsymbol{h}_u^k)$
$\boldsymbol{h}_G \leftarrow \mathtt{READOUT}(\{\boldsymbol{h}_v^K, \ \forall v \in V\})$
```

IV. Datasets

*1) **Physionet**:* Physionet dataset [19] contains EEG recordings of 197 subjects during their whole night sleep.[1] The dataset also contains EMG, EOG, body temperature, and respiration recordings. EEG signals have a sampling frequency at 100 Hz and each EEG trial was classified by well trained technicians with labels: W (awake), R (rapid-eye-moving sleep), 1, 2, 3 and 4, according to [20], however using channels Fpz-Cz/Pz-Oz EEGs instead of C4-A1/C3-A2 EEGs, as suggested by [21]. Each EEG trial has 2 channels and 3000 discretized time samples.

*2) **ErrP**:* The detection of error-related potentials (ErrP) to improve the accuracy of P300-based brain computer interface (BCI) speller [22].[2] The dataset was recorded from 16 healthy

[1] Physionet Dataset: https://github.com/XiaoxiWei/NeurIPS_BEETL

[2] https://www.kaggle.com/c/inria-bci-challenge/

subjects participating in an offline P300 spelling task. Spelling task had a fast mode (each item was flashed 4 times), and a slow mode (each item was flashed 8 times). Each subject performed 340 trials. If there is an inconsistency between subject's intention and BCI system, elicited ErrP should be detected. EEG data were recorded at a downsampled rate of 200 Hz from 56 channels. A trial has 250 discretized time samples, and is associated with a binary class label: erroneous (inferred item is different from the intent of subject) or correct feedback.

3) ***RSVP:*** A BCI system to type based on rapid serial visual presentation (RSVP) paradigm [23].[3] The dataset was collected from 10 healthy subjects, and consists of 41,400 trials of 16 channel EEG data. A g.USBamp biosignal amplifier with active electrodes was used to record trials during RSVP keyboard operations. Each trial has 128 discretized time samples, and is associated with one of the 4 labels: emotion elicitation, resting-state, motor imagery, or execution task.

V. Results & Discussion

We use PyTorch Geometric v2.0.2 [24] to implement GAT convolutional layers. We utilize stochastic gradient descent to train EEG-GAT framework instead of minibatch gradient descent, since the batching capacity of GAT layers is limited to a single graph due to the tensor manipulation framework which supports sparse matrix multiplication for rank-2 tensors [15]. A batch size of 1 graph strictly evaluates a universal representation for the matrix of attention weights invariant of subject identity or trial session. This allows assigning the same set of attention weights to every graph in the dataset and learning an edge weight matrix that is robust against the inter-subject and inter-session variations. All models were trained for 20 epochs on an NVIDIA Tesla K80 12GB GPU, and optimized by AdamW (Adam with adjustable weight decay) declared with an initial learning rate of 0.005, which exponentially decays at a rate of 0.9 every epoch.

ErrP and RSVP datasets were trained to observe EEG-GAT's within-subject performance, where we shuffle EEG trials from all subjects, and use the first 80% of EEG trials for training and the remaining 20% for validation. We utilize physionet dataset to observe EEG-GAT's cross-subject performance. Specifically, we use the EEG trials collected from subjects aged between 25-64 for model training and EEG trials collected from subjects aged between 65-79 for testing model's cross-subject performance.

Experimental results are outlined in Table II. EEG-GAT outperforms the existing GNN methods, especially for the RSVP dataset. While the classification accuracy of EEG-GAT is comparable to that of the state-of-the-art for ErrP dataset, we see EEG-GAT classifies only 58.09% of EEG trials in Physionet dataset. In general, classification accuracy of all GNN based models (GraphSage, Set2Set, SortPool, EdgePool, SagPool, GIN0 and EEG-GAT) is quite low compared to CNN based AutoBayes. The main problem that confronts us with the Physionet dataset is that it was recorded from only 2 channels and modeling a GNN for a graph with just 2 nodes is not an effective strategy.

TABLE II: Classification accuracy (%) results

Method	**Physionet**	**ErrP**	**RSVP**
GraphSAGE	54.49	74.44 ± 0.75	93.27 ± 0.05
Set2Set	55.33	75.38 ± 0.54	93.33 ± 0.09
SortPool	40.76	72.90 ± 0.61	93.24 ± 0.20
EdgePool	37.79	73.03 ± 0.96	92.89 ± 0.04
SagPool	41.53	74.41 ± 1.09	93.45 ± 0.19
GIN0	55.36	75.48 ± 0.60	93.26 ± 0.07
AutoBayes [25]	**63.02**	75.91 ± 0.44	93.42 ± 0.15
EEG-GAT	58.09	$\mathbf{76.42} \pm 0.29$	$\mathbf{96.27} \pm 0.18$

VI. Conclusion

The main contributions of this paper over the existing works are three-fold as follows:

- We extend the EEGNet, which is a de facto standard to extract frequency specific spatial features from EEG signals, by designing a neuroscientifically interpretable graph model, where node features are the EEG electrodes whose features extracted from the EEGNet backbone and edge weights between pairs of nodes are learned via taking advantage of the multi-head attention mechanism.
- While existing CNN and GNN based models are agnostic to the functional neural connectivity factor between electrode sites pertinent to a cognitive task, EEG-GAT can learn how the activity between different regions of the brain co-varies.
- Previous research in the literature use GAT convolutional layers to handle node classification in graph benchmark datasets such as: Cora, Citeseet and Pubmed. We extend GAT convolution to perform graph classification instead of node classification using various EEG datasets as a benchmark.

References

[1] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8459–8468.

[2] K. Huang, C. Xiao, L. M. Glass, M. Zitnik, and J. Sun, "SkipGNN: predicting molecular interactions with skip-graph networks," *Scientific reports*, vol. 10, no. 1, pp. 1–16, 2020.

[3] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The World Wide Web Conference*, 2019, pp. 417–426.

[4] Y. Yu, J. Chen, T. Gao, and M. Yu, "DAG-GNN: DAG structure learning with graph neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7154–7163.

[5] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[6] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, pp. 3844–3852, 2016.

[7] B.-H. Kim and J. C. Ye, "Understanding graph isomorphism network for rs-fMRI functional connectivity analysis," *Frontiers in neuroscience*, vol. 14, p. 630, 2020.

[8] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[3] http://hdl.handle.net/2047/D20294523

[9] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *arXiv preprint arXiv:1706.02216*, 2017.

[10] G. Dasoulas, J. F. Lutzeyer, and M. Vazirgiannis, “Learning parametrised graph shift operators,” in *International Conference on Learning Representations*, 2020.

[11] D. Cvetković and S. K. Simić, “Towards a spectral theory of graphs based on the signless Laplacian, II,” *Linear Algebra and its Applications*, vol. 432, no. 9, pp. 2257–2272, 2010.

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

[13] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, “EEGNet: A compact convolutional neural network for EEG-based brain–computer interfaces,” *Journal of neural engineering*, vol. 15, no. 5, p. 056013, 2018.

[14] A. Demir, T. Koike-Akino, Y. Wang, M. Haruna, and D. Erdogmus, “EEG-GNN: Graph neural networks for classification of electroencephalogram (EEG) signals,” in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, 2021, pp. 1061–1067.

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.

[16] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, “Connecting the dots: Identifying network structure via graph signal processing,” *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 16–43, 2019.

[17] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[19] B. Kemp, A. H. Zwinderman, B. Tuk, H. A. Kamphuisen, and J. J. Oberye, “Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the EEG,” *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 9, pp. 1185–1194, 2000.

[20] E. A. Wolpert, “A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects.” *Archives of General Psychiatry*, vol. 20, no. 2, pp. 246–247, 1969.

[21] B. Van Sweden, B. Kemp, H. Kamphuisen, and E. Van der Velde, “Alternative electrode placement in (automatic) sleep scoring (F_{pz}-C_z/P_z-O_z versus C_4-A_1),” *Sleep*, vol. 13, no. 3, pp. 279–283, 1990.

[22] P. Margaux, M. Emmanuel, D. Sébastien, B. Olivier, and M. Jérémie, “Objective and subjective evaluation of online error correction during P300-based spelling,” *Advances in Human-Computer Interaction*, vol. 2012, 2012.

[23] U. Orhan, K. E. Hild, D. Erdogmus, B. Roark, B. Oken, and M. Fried-Oken, “RSVP keyboard: An EEG based typing interface,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 645–648.

[24] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch Geometric,” in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[25] A. Demir, T. Koike-Akino, Y. Wang, and D. Erdogmus, “AutoBayes: Automated Bayesian graph exploration for nuisance-robust inference,” *IEEE Access*, vol. 9, pp. 39 955–39 972, 2021.