Music Information Retrieval from EEG data using Graph Volterra Filters

Jake Araujo-Simon
MEng ECE
ja858@cornell.edu

Fakharyar Khan
MEng CS
fsk36@cornell.edu

***Abstract*—We explore an application of graph-based signal processing and machine learning techniques to the modeling of brain dynamics as captured by EEG data generated during a music information retrieval task. Specifically, we study a graph-based extension of the Volterra series, known as the graph Volterra model (VGM) that exploits nonlinearity via Kronecker powers of the graph signal, which naturally live over a product graph. We implement a graph Volterra neural network (GVNN) based on the VGM and compare the results obtained to those using a standard graph neural network (GNN).**

I. Introduction

In recent years, there has been a growing interest in the application of graph-based methods to model and analyze brain activity. Ideally, one would like to be able to use non-invasive techniques such as an electroencephalogram (EEG) to measure brain activity in several regions in the brain, and attribute this multi-channel signal to an underlying graph representing some structural information about the brain. Of course, when using non-invasive methods like EEG one does not have direct access to the time-varying electrical potential being generated by the brain, but only the multi-channel signal measured at a set of electrodes. A fundamental problem is therefore formalizing the EEG signal as a graph signal in a meaningful way. There are various approaches to doing this; see, e.g., [1].

For example, one can use correlation between the signals obtained in pairs of regions over time as weights to form a weighted adjacency matrix that acts as a proxy for brain connectivity [2]. An edge with a large weight in this graph would then represent a tendency for two corresponding regions in the brain to be active and inactive at the same time which could potentially suggest that the two regions are coordinating to perform some task. Slightly more sophisticated perhaps, one can use the so-called 'coherence', defined as the squared norm of the cross-spectral density normalized by the product of the auto spectral densities of the two signals respectively [3].

On the other hand, substantial measurements in neuroanatomy have led to the development of the so-called 'structural connectivity matrix', which represents actual neuronal connectivity between brain regions, sometimes referred to as the brain 'connectome'. See e.g. [4]. This is very appealing in principle since one can then obtain a basis of Laplacian eigenfunctions over the graph, allowing for spectral decomposition of brain signals. However, apparently there can be statistically significant variation in how the graph is constructed; see [5]. Additionally, even if one has access to a good structural-connectivity matrix, it is not immediately appear (to us, yet, in any case), how to associate the EEG signal to nodes of this graph, since the electrodes are physically separated from the brain itself (and may pick up contributions over an entire region, etc).

In the following, we describe how to first obtain an underlying graph and graph signal from EEG data, and then develop a signal processing/machine learning algorithm in order to model, predict and classify brain signals. In particular, we investigate the application of Volterra series defined over graphs, focusing on so-called graph Volterra models, as well as briefly discussing their topological generalization, topological Volterra filters, as described in [6-7]. Volterra series have recently received attention in a machine learning context (see [8-9]), although to our knowledge Volterra Neural Networks have not yet been defined in full generality over graphs.

In this paper, we implement a basic graph Volterra neural network, as well as a standard graph convolutional neurel network GCNN for comparison. In order to train a Volterra filter, we initially attempted to use a naive least-squares based approach followed by a linear classifying layer using the one-hot encoded signals corresponding to the class labels. However, despite obtaining high accuracy on an artificial regression problem, this didn't seem to work: test set accuracy remained low, and training set accuracy plateaued. Thus, we switched over to a neural network based approach, using gradient descent with backpropagation (implemented in PyTorch) to learn the Volterra filters directly with the classification as the final layer and cross-entropy as the loss function.

Thus, we aim to apply graph-based Volterra models to analyze and make inferences on graph data representing brain activity. To test our algorithm, we plan to use the OpenMIIR dataset which is a public dataset of EEG recordings taken in a study in which subjects were instructed to imagine (i.e., audiate internally) one of 12 short music fragments (7-16 seconds) that they had previously heard [13]. Building on the dataset's original purpose, our goal will be to determine whether a trainable graph Volterra filter can be used to reliably decode the specific music fragment that a subject is thinking of.

II. Problem Formulation

The problem formulation is as follows. First, let $L^2(\mathbb{R}) = L^2(\mathbb{R}, \mathbb{C})$ denote the Hilbert space of square-integrable, complex-valued signals over the real line. Then, given a set of multichannel signals, $X = \{\boldsymbol{x}_i\}_{i \in I}$, one for each subject in a set of I subjects, where $\boldsymbol{x}_i : (L^2(R))^K$ is the per-subject set of k signals measuring the time-varying electrical potential of the user's brain over k different electrodes, the problem is to determine a mapping from brain activity into the target classifying space, which in our case is the set of songs the participants listened to, that matches the observed data.

III. Proposed Approach

A. Preprocessing

In order to make the OpenMIIR dataset suitable for graph-based analysis, a variety of preprocessing measures were taken. First, for each subject we separated each trial (where they were asked to imagine of the 12 music fragments) into five second intervals. The main motivation behind this was that because we had planned on applying methods like GCNNs and Volterra networks which can be very expressive and "data hungry", there is a risk of the model overfitting very easily if the dataset is too small. Furthermore, given that the length of a trial ranged from 7-12 seconds and we were more interested in the simultaneous activation of different regions in the brain instead of the actual activity, we believed that five seconds would be more than sufficient to capture that information.

After segmenting each trial into five-second snippets, the EEG data was then filtered using a band pass filter so that the frequency components outside of the 8-12 Hz range were significantly attenuated. EEG data can be very noisy and can often contain artifacts caused by muscle activity such as eye blinking. Furthermore, studies have found that most brain activity operates within a very low frequency range in which different frequency bands correspond strongly to different types of brain activity. For example, the delta band ranges from 0.5-4Hz and is strongly associated with fairy low activity where the subject is either in deep sleep or unconscious. For our purposes, we were primarily interested in the alpha band (8-12Hz) which is associated with relaxed wakefulness and calmness and is typically the band that brain activity largely resides in when a person is listening to music [14]. Filtering for only this range of frequencies allowed us to extract out meaningful and relevant information from the raw EEG data and significantly reduced noise.

After doing this, we then calculated the coherence between each pair of signals that were measured from the 64 nodes and obtained a graph in which the nodes were the 64 regions whose brain activity were measured and the weight of an edge between two nodes was the coherence measured between the two corresponding regions.

Originally, we had considered using the coherence matrix as the only input to the model and not include the actual EEG signal from each node. However, in practice, when we trained a graph convolutional classifier on our dataset, we found that the model had performed somewhat poorly on the test set and so we decided to include the EEG signal to provide our models with more information on the patient's brain activity. Since we had filtered each EEG signal to only contain frequencies from 8-12Hz, to prevent aliasing, we sampled the signal at a rate of 24Hz. With this, each node in our graph contained a feature vector with 120 components that described the brain activity in the corresponding region within that five-second interval and this gave us a much more comprehensive representation of the user's overall brain activity.

B. Graph Convolutional Neural Network

Similar to how convolutional neural networks (CNNs) cascade learnable filters that are applied to an input image in series, a graph CNN interweaves a series of linear graph filters and nonlinear activation functions to create a neural network that can extract rich features from a graph which in turn can be used for tasks such as classification [15]. In particular, let x be some signal indexed over a graph. In our case, x is a matrix of 69x120 size in which each row represents the sampled EEG signal measured from each node. Then a single graph convolutional layer of order K would look like

$$\phi(X) = \sigma(\sum_{k=0}^{K-1} h_k S^k x)$$

where σ is some non-linear activation function, S^k is the graph shift operator applied k times, and h_k are the learnable filter coefficients. We can then concatenate multiple layers together to create a more expressive model. Additionally, in order to allow our model to learn multiple sets of features in a single layer, we can create multiple channels in which multiple filters are applied to the input graph signal in parallel. Then after the convolution layers, the output is typically pooled and then flattened and passed through a fully connected neural network for whatever task the model is being trained for.

C. Construction of the Volterra filter

The Volterra series is a universal model of nonlinear systems with fading memory of the form:

$$y(t) = V(s)(t) = \sum_{j=0}^{\infty} V_j(s)(t)$$

where the $V_j : L^2(\mathbb{R}) \to L^2(\mathbb{R})$, termed *homogeneous Volterra operators*, are defined by:

$$y_j(t) = V_j(s)(t) = \int_{\boldsymbol{\tau}_j \in \mathbb{R}^j} v_j(\boldsymbol{\tau}_j) \prod_{r=1}^{j} s(t - \tau_r) d\boldsymbol{\tau}_j \qquad (1)$$

Volterra series have been generalized to the graph case, via so-called graph Volterra models, in a number of works [6,7], the last of which presented a fully-general, topological model capturing both multi-hop interactions over the network (analogous to the general case of graph filters), and higher-order interactions (analogous to those captured, for example,

by techniques from simplicial signal processing using filters defined via the Hodge Laplacian [11-12]).

The authors in [6] define a graph Volterra filter at each order. First, recall that a simple one-hop graph filter is of the form:

$$\boldsymbol{y} = \boldsymbol{S}_1 \boldsymbol{x} \tag{2}$$

which filters the graph signal **x** by the GSO, $\boldsymbol{S}_1$. Generalizing to multiple hops, we obtain a traditional graph filter that is given as a polynomial in the GSO:

$$\boldsymbol{y} = \sum_{k=0}^{K} h_k \boldsymbol{S}_1^k \boldsymbol{x} \tag{3}$$

Note that the filtering process is still linear in the graph signal, since the polynomial linearity is applied only to the GSO itself. This is equivalently a 1st-order topological Volterra filter, analogous to how a first-order Volterra series is a linear time-invariant (LTI) system.

As explained in [6], the above linear graph filter structure captures one-hop dynamics, and can be generalized and extended in two different directions: by using multiple hops (as in traditional, multi-hop graph filters); and by including higher-order interactions (generalizing from pair-wise interactions to k-ary ones). Although we limit ourselves in this study to the graph case, we present the fully-general topological definitions here for context. The distinction the authors in [6] make between so-called graph Volterra filters and topological Volterra filters corresponds to the cases of: single-hop, k-ary interactions (graph Volterra models); and multi-hop, k-ary interactions (topological Volterra filters).

As in conventional Volterra series for signals which are functions of a time variable $t \in \mathbb{R}$, at higher orders the Volterra operators model nonlinearity as a filtering process of tensor (or in the discrete case, Kronecker products) products of the input signal with itself. Thus, at order 2, the graph Volterra and topological Volterra filters are defined, respectively, as:

$$\boldsymbol{y} = \boldsymbol{S}_2 \boldsymbol{x}^{\otimes 2} \tag{4}$$

and

$$\boldsymbol{y} = \sum_{k=0}^{K} \sum_{l_1,l_2=0}^{L_1,L_2} h_{k,l_1,l_2} \boldsymbol{S}_1^k \boldsymbol{S_2} (\boldsymbol{S_1}^{l_1} \otimes \boldsymbol{S_1}^{l_2}) \boldsymbol{x}^{\otimes 2} \tag{5}$$

where the h_{k,l_1,l_2} now correspond to a 2-dimensional function called the (2nd-order) *graph Volterra kernel function* (GVKF), and the matrix $\boldsymbol{S}_2 : \mathbb{R}^{N \times N^2}$ is an incidence matrix that plays the role of higher-order GSO, and describes the influence of each node pair on each individual node. Generalizing this same pattern to arbitrary dimension, we obtain the fully-general expression for the N^{th}-TVF:

$$\boldsymbol{y} = \sum_{k=0}^{K} \sum_{l_1,\dots l_P=0}^{L_1,\dots L_p} h_{k,l_1,\dots l_p} \boldsymbol{S}_1^k \boldsymbol{S_p} (\boldsymbol{S}_1^{l_1} \otimes \cdots \otimes \boldsymbol{S}_1^{l_p}) \boldsymbol{x}^{\otimes p} \tag{6}$$

In the next section, we describe a simplified neural network structure based on the concept of graph Volterra model.

D. Graph Volterra Neural Network

Slightly re-interpreting the graph Volterra model structure defined in section 3.C, we designed a second-order GVNN as follows. First, we took the first-order shift operator, S_1 to be the graph Laplacian of the coherence graph defined above. We then defined the second-order shift operator, S_2, simply as the second Kronecker power of the underlying graph Laplacian, $S_2 = L^{\otimes 2}$. Note that in this case we can effectively compute the eigenbasis, i.e. graph Fourier basis, of S_2 simply as the Kronecker product $U \otimes U$ of the matrix U of eigenvectors of the underlying graph Laplacian. The full output of the graph Volterra model is then:

$$\boldsymbol{y} = \boldsymbol{h_0} + F_1^{-1}(\hat{\boldsymbol{h}}_{\boldsymbol{1}} \odot \hat{\boldsymbol{x}}) + \text{diag}(F_2^{-1}(\hat{\boldsymbol{h}}_{\boldsymbol{2}} \odot \hat{\boldsymbol{x}}^{\otimes 2})) \tag{7}$$

where F_1 is the GFT of the underlying coherence graph, F_2 is the GFT of the second Kronecker power of the same, and $\text{diag}(-)$ denotes the operation of slicing along the diagonal in G^2, i.e. along pairs of the form (v, v) for all vertices v. Note that in this model there are no separate activation functions, rather the graph Volterra kernel functions and higher-order graph signals themselves directly encode the nonlinear features.

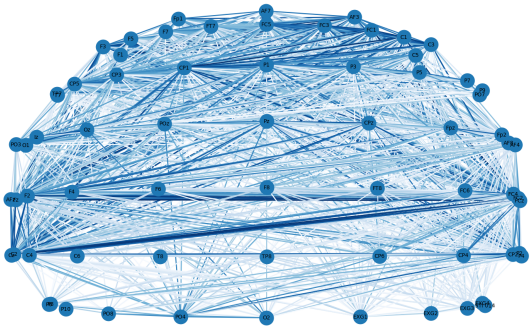
IV. Analysis and Numerical Approach

A. Validity of Coherence Graph Representation

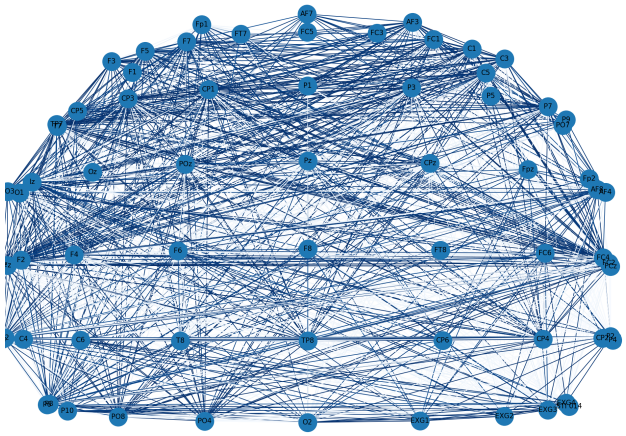
One of the first design choices that we came across in this project was whether to use correlation or coherence as a measure of coordination between regions in the brain. To determine this, we computed the adjacency matrix computed for one of the trials using each measure and plotted the corresponding connectivity graphs as can be seen in Figure 1.

From this plot, we found that the connectivity graph produced using correlation was fairly noisy and didn't seem to indicate any significant collaboration between different regions. On the other hand, when we used the coherence between different regions as the edge weights, we saw that the edge weights followed a more bimodal distribution in which the majority of edges either had fairly large or small weights and there very few that were in between. Because of this, we believed that the coherence measure was much more effective at discriminating and determining which pairs of regions in the brains were in coordination and which weren't.

For further validation, we also sampled some of the pairs of nodes that had very strong edges between them and checked to see if the corresponding parts of the brain performed similar functions or were known to collaborate with one another for certain tasks. For example, there is a very strong edge between the C4 and the FC2 nodes. Both nodes are generally involved with tasks such as motor planning (which can involve things such as determining the sequence of actions necessary to perform a motor action such as grabbing a cup). Additionally, we observed that there was a fairly strong edge between nodes P1 and P3. The P1 node is usually associated with



(a) Connectivity Graph Produced Using Coherence



(b) Connectivity Graph Produced Using Correlation

Fig. 1: Comparison of Connectivity Graphs

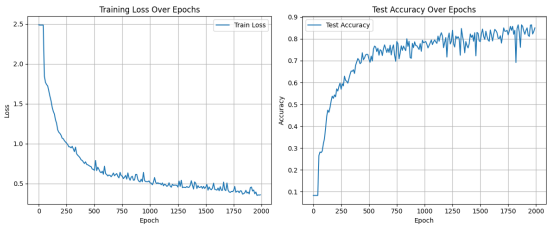
sensory integration which involves processing and organization sensory information such as sound. Furthermore, the P3 node is typically associated with "context updating" which is when your brain processes and responds to sudden changes in the environment [16]. Both of these nodes having a strong coherence with one another makes sense because right before each trial, the patients were asked to listen to one of the twelve songs that they would then imagine in their head.

B. Model Performance

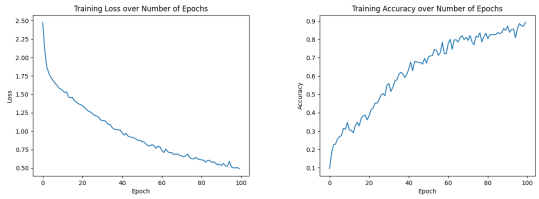
To serve as a benchmark to compare our Volterra neural network against, we trained a Graph Convolutional Neural Network (GCN) using a single-channel input, composed of four convolutional layers with first-order linear filters followed by two fully connected linear layers with a hidden layer size of 128. After 2000 epochs of training with a learning rate of 1e-3 and the AdamW optimizer[17], we were able to achieve a test accuracy of roughly 86.39 percent. Beyond just serving as a benchmark, the effectiveness of our relatively simple model also shows the power of modeling the patients' brain activity as a graph and motivates our exploration of more expressive models such as the Volterra filter.

For our Volterra neural network, we used a single Volterra filter as described earlier with learnable coefficients and then, just as with the GCN, we fed the output into an MLP with two fully connected linear layers each with a hidden layer size of 128. Due to the computational and space complexity of the forward pass of the Volterra filter, we subsampled both the graph's nodes and the graph signal by taking the 30 nodes with the largest degree centrality measure and average downsampling the graph signals by a factor of 4 respectively. Despite needing to do this, after training for only 100 epochs, our model was able to achieve a test accuracy of 94.3 percent as shown in Figure 2.

One thing to note is that because we used the same size and number of linear layers in both models and the same hyperparameters and learning optimizer, the difference in performance between the two models can be attributed almost entirely to the effectiveness of the Volterra filter for feature construction. Furthermore, despite using 4 graph convolution layers, our volterra neural network with a single Volterra convolution layer was able to outperform the GCN not only in test accuracy but also in the speed at which it converged to the optimal solution (100 vs 2000 epochs).



(a) GCN model performance (Loss and Accuracy).



(b) Volterra model training performance (Loss and Accuracy).

Fig. 2: Comparison of GCN and Volterra model performance over number of epochs.

V. Conclusion

In this paper, we have explored the application of graph-based signal processing and machine learning techniques for the task of music information retrieval from EEG data. Towards this goal, we have developed a graph volterra neural network (GVNN) that uses second-order volterra filters in place of linear graph filters for feature construction and demonstrated the ways in which it outperforms a regular graph convolutional network.

References

[1] *Brain connectivity estimators*. https://en.wikipedia.org/wiki/Brainconnectivityestimators

[2] *Connectivity Analysis in EEG Data*.Encyclopedia MDPI,2023. https://encyclopedia.pub/entry/46264

[3] *Coherence (signal processing)*. https://en.wikipedia.org/wiki/Coherence(signalprocessing)

[4] K.Glomb*et al.*, "Connectome spectral analysis to track EEG task dynamics on a sub-second scale," *NeuroImage*,vol.221,2020. https://www.sciencedirect.com/science/article/pii/S1053811920306236

[5] S. I. Dimitriadis *et al.*, "Graph Laplacian Spectrum of Structural Brain Networks is Subject-Specific, Repeatable but Highly Dependent on Graph Construction Scheme," *bioRxiv*, 2023. https://www.biorxiv.org/content/10.1101/2023.05.31.543029v1

[6] M. Yang, E. Isufi,G. Leus, "Topological Volterra Filters," in *Proc. ICASSP 2021*. https://ieeexplore.ieee.org/document/9414275

[7] Xiao et al., Distributed Nonlinear Polynomial Graph Filter and Its Output Graph Spectrum: Filter Analysis and Design, https://ieeexplore.ieee.org/document/9343697

[8] S. Roheda, H. Krim, B. Jiang, "Volterra Neural Networks (VNNs)," *Journal of Machine Learning Research*, vol. 25, pp.1–29, 2024. https://www.jmlr.org/papers/volume25/21-1082/21-1082.pdf

[9] M. Banerjee *et al.*, "VolterraNet: A Higher-Order Convolutional Network with Group Equivariance for Homogeneous Manifolds," preprint, 2021. https://rudra1988.github.io/articles/volterra.pdf

[10] Robustness of connectome harmonics to local gray matter and long-range white matter connectivity changes, https://www.sciencedirect.com/science/article/pii/S1053811920308508

[11] Anand et al., Hodge Laplacian of Brain Networks, https://pmc.ncbi.nlm.nih.gov/articles/PMC10909176/

[12] S. Dakurah *et al.*, "Modelling Cycles in Brain Networks with the Hodge Laplacian," in *Proc. MICCAI 2022*. https://pages.stat.wisc.edu/~mchung/papers/dakurah.2022.MICCAI.pdf

[13] S. Stober *et al.*, "Towards Music Imagery Information Retrieval: Introducing the OpenMIIR Dataset of EEG Recordings from Music Perception and Imagination," ISMIR 2015. https://github.com/sstober/openmiir

[14] *Neural oscillation*. Encyclopaedia Britannica, updated 15 Mar 2025. https://www.britannica.com/science/brain-wave-physiology

[15] T. N. Kipf, M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," arXiv:1609.02907, 2016. https://arxiv.org/abs/1609.02907

[16] *Electroencephalography*, https://en.wikipedia.org/wiki/Electroencephalography

[17] *AdamW — PyTorch 2.7 documentation*, https://docs.pytorch.org/docs/stable/generated/torch.optim.AdamW.html