# BrainMixup: Data Augmentation for GNN-based Functional Brain Network Analysis

Alexis Li*
Hamilton High School
Gilbert, United States of America
li.alexis1111@gmail.com

*Abstract*— **Different data augmentation techniques such as mixup have been applied to Graph Networks to reduce overfitting. Many graph mixup techniques have complicated implementations to deal with graph data's non-Euclidian data structure. However, not many papers have explored applying traditional mixup techniques to brain data that has the unique capacity in that nodes are defined in place by their ROIs. Thus, this paper proposes BrainGMixup, applying previously proposed mixup techniques meant for 1-D feature vectors to 2-D graph node features and edge index matrices. The results and experiments in this paper demonstrate that applying this simple modified mixup increases a model like EdgeConv' s ability to generalize on the training data and boosts its performance on a variety of metrics.**

## I. Introduction

The human brain is a complex made of 86 billion neurons- nearly the number of stars in the Milky Way- with an estimated quadrillion synapse. Even now, scientists are still analyzing these connections to map the function and mechanisms of the human brain all of which is critical to battling neurodegenerative disorders such as Alzheimer's disease and gaining a new understanding of mental disorders such as attention deficit hyperactivity disorder (ADHD), schizophrenia, and autism spectrum disorder (ASD).

Graph Neural Networks (GNNs) have been at the center of non-Euclidian data analysis with applications in molecule interaction prediction, social media recommenders, and drug discovery [1]. As of late, GNNs have been applied to brain connectome analysis which is key to advancing deep learning classification and identification of brain related conditions/disorders.

Previous works have explored attention based GNNs, different methods of node feature extraction, pooling methods, etc. One area of particular interest is developing different methodologies to combat the issue of overfitting and network memorization over generalization as graph data is often noisy and low in quantity. Different data augmentation techniques to mitigate these issues involve neighborhood level feature generation [2], edge modification/removal [3], and consistency regularization [4]. However, many of these methods end up losing important node information by dropping them out or don't fully engage with the uniqueness of brain connectome data. Insofar that brain connectome data doesn't face issues of graph isomorphism as pre-determined regions of interest (ROIs) dictate the position of nodes, it is possible to explore previous data augmentation methods for Euclidean data-based models.

The motivation of this paper is to explore the application of a modified version of mixup based on the Vicinal Risk Minimization principle [5] on an EdgeConv GNN for ASD classification. It is important to note that variations of mixup for Graph Neural Networks have been applied in previous papers [6], but they deal with graph datasets that have fluctuating numbers of nodes that come in different order that require far more complex implementations. The form of mixup proposed in this paper allows it to be utilized on 2-D node feature and edge index matrices rather than just 1-D feature vectors that are typical for image data types.

## II. Related Works

**Graph Mixup.** Many Graph Mixup methods involve constructing new synthetic graphs from probability matrices to connect new sampled subgraphs to the original graph or to reorder/mixup the original graph. All these methods also seek to develop methods to combat graph data irregularity and keeping parts of original graph structure. G-Mixup [6] applies the principles of mixup to probability matrices with individual points representing the likelihood of an edge existing between two nodes. These graphons then allow for the creation of synthetic graphs that retain properties of the original graph and the new graphon. Graph Transplant [13] seeks to sample the top salient nodes in a graph to retain the original structure and then append a partial K-hop subgraph using node features to predict edge existence.

However, brain data relies on preset ROI locations for all nodes with a fixed structure so sampling subgraphs from other regions of the brain and combining them with other sections becomes more complicated when considering that the node features are the correlation matrix measures between specific node connections. Additionally, brain graph data's fixed nature allows it to become more malleable to typical image classification mixup techniques.

*This work was done under the mentorship of Dr. Carl Yang from Emory University (j.carlyang@emory.edu).

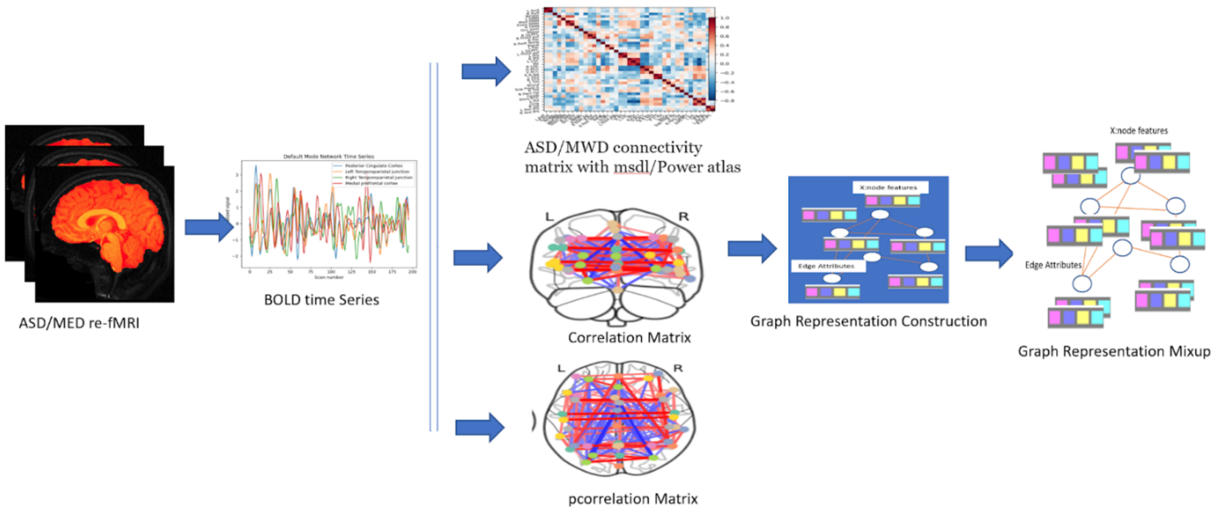
ASD/MED re-fMRI
BOLD time Series
ASD/MWD connectivity matrix with msdl/Power atlas
Correlation Matrix
pcorrelation Matrix
Graph Representation Construction
X:node features
Edge Attributes
Graph Representation Mixup

Fig 1. Graph Data Construction

## III. Proposed Models

### A. Graph Data Construction

Preprocessed functional MRI (fMRI) ASD data is fetched from the ABIDE dataset using the nilearn pipeline [7] for 871 subjects. Functional MRI data is obtained by measuring changes in blood flow over time in response to some kind of stimulation. This dataset includes time series data, multiple scans for every patient at different points of time, but this paper does not utilize them.

The full graph data construction process is shown in Figure 1. The MSDL atlas from nilearn was used to determine 39 ROIs for node placement. Each node had a correlation matrix and partial correlation matrix extracted with the partial correlation matrix representing the top 10 neighbors for a particular node with the top 10 neighborhoods being labeled with 1 and the others labeled with 0. The coefficient for the correlation between two ROIs goes from 1 to -1 with 1 indicating high correlation and -1 indicating inverse correlation. The correlation matrix being calculated as the average for the ROIs coefficients over a time series.

The correlation matrix served as the node features with the edge indexes and weights being derived from the partial correlation matrix. ABIDE also contains graph wide information regarding the subject but individual nodes only have connectivity matrix information to work off of. Edge attributes were derived from the partial correlation matrix with all values of 1 replaced with their original correlation matrix values.

A second dataset based in measuring the response of children from the ages of 3-12 and adults to a film for 155 subjects was also utilized with preprocessed T1W BOLD data [10]. Clinical data regarding gender, handedness, and age was also provided. Subjects watched a short movie, Pixar's Partly Cloudy, while fMRI scans were taken. No task was given. The classification task for the network was a binary age classification (child/adult) prediction. Graph data was generated in the same process that ABIDE was with 39 node features for every node. This paper will refer to this dataset as the Movie Watching Development dataset (MWD).

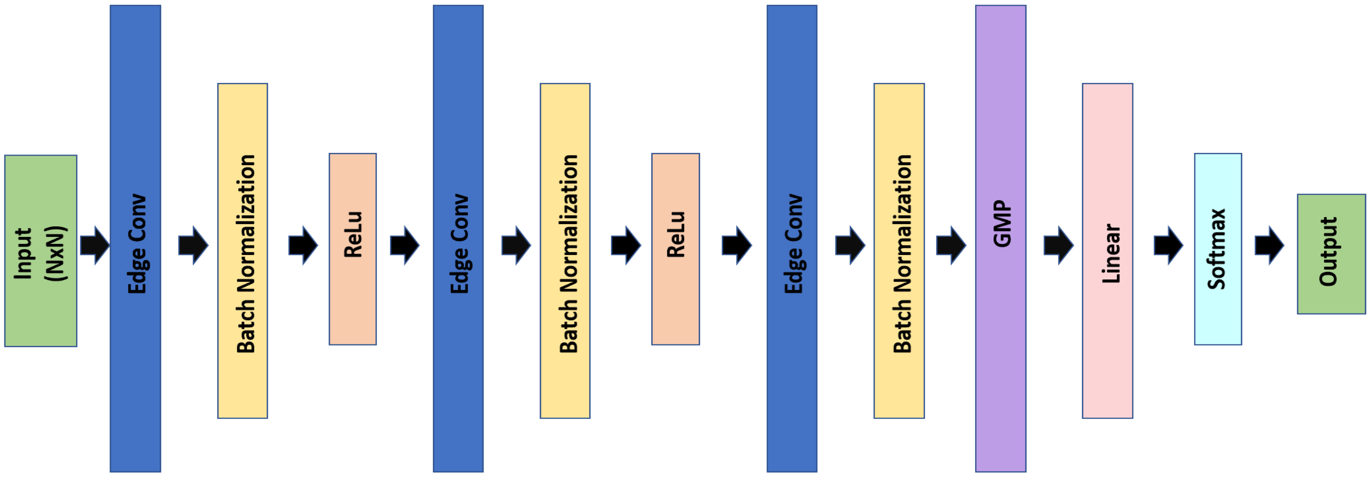

Fig 2. Edge Conv Network

## B. *EdgeConv Network*

The full EdgeConv Network is displayed in Figure 2. In an EdgeConv layer, edge features as defined by $e_{ij} = h_\Theta(x_i, x_j)$ with $h_\Theta: \mathbb{R}^F \times \mathbb{R}^F \rightarrow \mathbb{R}^{F'}$ functioning as the MLP for the model further defined by a non-linear function parametrized by a set of learnable parameters [8]. In this equation, $x_i$ represents the embedding of node i with $x_j$ representing the embeddings of all of node i's neighbors. The neighbors also include the node itself. In this case, a mean aggregation operation is performed for all the edge features to get the final embedding for the node' and its neighbor's edges represented by the following equation:

$$x_i' : \underset{j_i(i,j)\in\varepsilon}{mean}\, h_\Theta(x_i || x_j - x_i) \quad (1)$$

This type of method allows for the extraction of neighborhood level features and grouping of different nodes based on nearest neighbor processing in data construction. Edge Conv also allows for different aggregation methods such as the mean across the embeddings of the node and its neighbors.

For this model, three EdgeConv layer were applied with batch normalization being applied for each layer and ReLu applied for the first two. One embedding was created using a global mean pool for the entire batch with a linear layer then Softmax applied. This embedding is used to do classification for two classes for the network.

The graph is also dynamically updated by calculating the pairwise distance matrix for the features and taking the closest k neighbors for every point. Different variations of k were used in this paper's experiments to test the level of graph construction needed to obtain desirable results.

## C. *MixUp*

Remedying issues of overfitting caused by memorization in deep learning models has been a concern to all within the field. Issues of lack of viable data have made it difficult in the areas of brain analysis to gain proper results and proper methods of developing truly synthetic have yet to be standardized.

A method proposed utilizes a vicinal distribution labeled mixup principle during training which reduces the empirical vicinal risk rather than reducing empirical risk [5]. This method has been proven to increase a model's generalization capability without generating computational stress by creating data within the vicinity of given data points. However, this method was initially created for 1-D feature input vectors which doesn't match most Graph Network datasets that input 2-D node feature matrices.

Methods that have adapted mixup for Graph Networks have more complicated implementation than what is needed for brain connectome graph analysis as node placement is fixed in ROIs. To sidestep this issue, this paper proposes a modified version of mixup that applies the function across all rows for both the node feature matrix and edge index matrix:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j \ \ (2) \ \text{ where i,j} = 1, \dots, \text{N}, i \neq j$$

$$\widetilde{ei} = \lambda ei_i + (1-\lambda)ei_j \ (3) \ \text{where i,j} = 1, \dots, \text{N}, i \neq j$$

$$\tilde{y} = \lambda y_i + (1-\lambda)y_j \quad (4)$$

$N$ representing the number of rows in a 2-D, $x$ representing the node feature matrix, $ei$ for the edge index matrix, and $y$ as the labels. $\tilde{x}$, $\tilde{y}$, $\widetilde{ei}$ are post-mixup versions of their above matrices. Thus, all rows and all columns in a 2-D matrix have been modified.

$$l = \lambda \cdot c(p, y_a) + (1-\lambda) \cdot c(p, y_b) \quad (5)$$

The mixup loss criterion used from [5] uses the predicted value ($p$), a defined criterion function ($c$), and the mixup labels from the original mixup data function with $y_a$ as the original label and $y_b$as the label from the data mixed in. Using a regular criterion function would fail to factor in the new mixed y labels and would thus muddle the training of the network.

Mixup as a data augmentation technique is relatively easy to implement and further encourages the model to behave in a more linear manner and adjust to many different scenarios rather than just the original training data. This also helps to prevent training accuracy and metrics from drastically increasing within a few epochs without the same effect in the test accuracy.

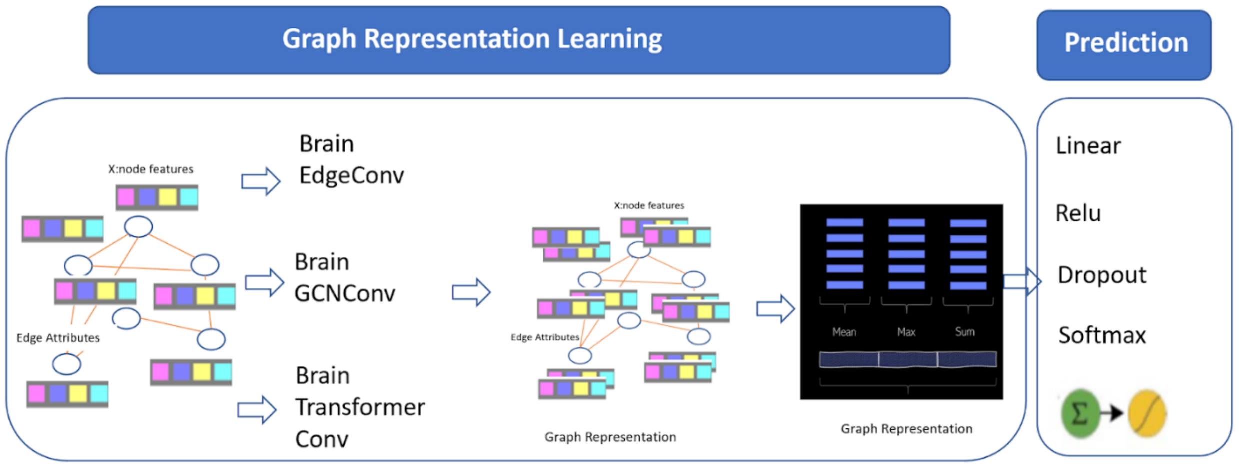

Fig 3. Graph Representation Learning and Prediction

## IV. Results/Experiments

| Model | Mixup | Development Data | | | | |
|---|---|---|---|---|---|---|
| | | Accuracy (%) | Sensitivity | Specificity | Recall | F1 |
| TransformerConv | No | 0.90 | 0.50 | 1.00 | 0.50 | 0.50 |
| GCNConv | No | 0.90 | 0.71 | 0.96 | 0.71 | 0.77 |
| EdgeConv | No | 0.90 | 0.83 | 0.92 | 0.83 | 0.77 |
| | Yes | 0.94 | 0.67 | 1.00 | 0.67 | 0.80 |

Table 1. MWD Results

| Model | Mixup | ABIDE Data | | | | |
|---|---|---|---|---|---|---|
| | | Accuracy (%) | Sensitivity | Specificity | Recall | F1 |
| TransformerConv | No | 0.63 | 0.63 | 0.62 | 0.68 | 0.63 |
| GCNConv | No | 0.54 | 0.60 | 0.45 | 0.60 | 0.59 |
| EdgeConv | No | 0.58 | 0.67 | 0.47 | 0.67 | 0.64 |
| | Yes | 0.68 | 0.85 | 0.48 | 0.92 | 0.85 |

Table 2. ABIDE Results

### A. *Comparative Models*

Figure 3 displays the comparative models and EdgeConv. The results of Edge Conv are compared with those of a Transformer Convolution Network (TransformerConv) [11] and a Graph Convolutional Networks (GCNConv) [12]. The TransformerConv network utilizes three attention heads across 4 different layers with every layer having batch normalization applied. The final x representation from a global mean pool function goes through two cycles of ReLu on a linear layer with dropout before applying Softmax for the final output. The GCNConv network contains 3 layers with a convolution layer and ReLu and dropout applied.

### B. *Performance comparison*

Starting off by focusing on all models that did not use mixup, EdgeConv without mixup is seen to be comparable to both GCNConv and TransformerConv accuracy wise with the highest sensitivity for both ABIDE and MWD data. Recall is also higher for EdgeConv without mixup regarding MWD and EdgeConv' s F1 is the highest for ABIDE data. This seems to indicate that for MWD data all models perform relatively similarly which is not as surprising given the smaller data quantity. For ABIDE data, TransformerConv has all the highest metrics, except in sensitivity although not by much, which indicates that focusing on the most important nodes helps ease overfitting absent other data augmentation techniques.

EdgeConv with mixup outperforms all models on nearly every metric. For MWD data, EdgeConv with mixup sees a 4% increase in accuracy over all other models with one of the highest specificities and f1 scores. For ABIDE data, it sees a 5% increase in accuracy compared to the second highest accuracy model, TransformerConv, with increase in sensitivity, f1, and especially recall. Regarding EdgeConv accuracy, MWD data saw a similar 4% increase and ABIDE data saw a 10% increase with mixup. All of these metrics point to mixup's ability to increase generalization and number of correctly identified positive patients for models.

## V. Conclusion

This work allows for the application of mixup on 2-D brain graph data, taking advantage of brain data fixed node structure and mixup's simplistic implementation to boost a model's generalization ability. This was shown through an increase in accuracy for both MWD and ABIDE data for EdgeConv that surpassed those of other models such as TransformerConv and GCNConv. A data augmentation technique tailored to brain graph data will provide a simple method to creating more robust models for further experimentation. This also leads into future exploration into creating more learnable brain mixup techniques that rely less on random values covering a wide breadth of vicinity graphs and more on selection based off of what previously had not been covered.

## ACKNOWLEDGEMENT

I thank Dr. Carl Yang, Mr. Xuan Kan, Ms. Hejie Cui and Mr. Wei Dai from Emory University for their ideas, mentorship and technical assistance.

## REFERENCES

[1] S. Ivanov, "Top applications of graph neural networks 2021," Criteo R&D Blog, Jan. 17, 2021. https://medium.com/criteo-engineering/top-applications-of-graph-neural-networks-2021-c06ec82bfc18

[2] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah, "Data augmentation for graph neural networks," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 12, pp. 11015–11023, May 2021, doi: 10.1609/aaai.v35i12.17315.

[3] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: towards deep graph convolutional networks on node classification," openreview.net, Mar. 11, 2020. https://openreview.net/forum?id=Hkx1qkrKPr (accessed Oct. 12, 2022).

[4] H. Park et al., "Learning Augmentation for GNNs With Consistency Regularization," IEEE Access, vol. 9, pp. 127961–127972, 2021, doi: 10.1109/ACCESS.2021.3111908.

[5] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," arXiv:1710.09412 [cs, stat], Apr. 2018, [Online]. Available: https://arxiv.org/abs/1710.09412

[6] X. Han, Z. Jiang, N. Liu, and X. Hu, "G-Mixup: Graph Data Augmentation for Graph Classification," arXiv:2202.07179 [cs], Feb. 2022, Accessed: Oct. 12, 2022. [Online]. Available: https://arxiv.org/abs/2202.07179

[7] "nilearn.datasets.fetch_abide_pcp," Nilearn. https://nilearn.github.io/stable/modules/generated/nilearn.datasets.fetch_abide_pcp.html (accessed Oct. 12, 2022).

[8] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," ACM Transactions on Graphics, vol. 38, no. 5, pp. 1–12, Nov. 2019, doi: 10.1145/3326362.

[9] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik, "Vicinal Risk Minimization." Accessed: Oct. 14, 2022. [Online]. Available: https://papers.nips.cc/paper/2000/file/ba9a56ce0a9bfa26e8ed9e10b2cc8f46-Paper.pdf

[10] "OpenNeuro," openneuro.org. https://openneuro.org/datasets/ds000228/versions/1.0.0 (accessed Oct. 14)

[11] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, "Masked label prediction: unified message passing model for semi-supervised slassification." Accessed: Oct. 16, 2022. [Online]. Available: https://arxiv.org/pdf/2009.03509.pdf

[12] T. Kipf and M. Welling, "Semi-Supervised classification with graph convolutional networks" [Online]. Available: https://arxiv.org/pdf/1609.02907.pdf

[13] J. Park, H. Shim, and E. Yang, "Graph transplant: node saliency-guided graph mixup with local structure preservation," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, no. 7, pp. 7966–7974, Jun. 2022, doi: 10.1609/aaai.v36i7.207