Graph Convolutional Networks for Fly Connectome Data

Anvita Gupta
Stanford University
Stanford, CA 94305
avgupta@stanford.edu

Shreya Shankar
Stanford University
Stanford, CA 94305
shreya1@stanford.edu

Abstract

Recent advances in microscopy technology have made it possible, for the first time, to assemble high quality complete connectomes of organisms. However, one major challenge neuroscientists face is predicting functions of neurons given only their structural information. In this project, we apply deep learning approaches to bridge this gap between structure and function, using the Janelia FlyEM dataset. We apply deep graph convolutional neural networks (GCNs) to infer functional classes of neurons, given their locational data in the fly visual system. We furthermore apply a semisupervised learning approach, aiming to label neurons given only a very small subset of training data. Given 25% of the available data, our GCN predicts a neuron's functional class out of fourteen possible classes, and achieves an accuracy of 74%– which is significantly better than random. This approach can help guide neuroscientists in searching connectome data for neuronal circuits that conduct different functions in biology.

1 Introduction

The fruit fly brain is ideal to choose for connectome analysis because this relatively small brain has a high level of complexity. Current genetic tools allow scientists to extract detailed features from fruit fly brain data and compare individual neurons on the basis of structure, and long term, on the basis of function. The fly brain has around 100,000 neurons.

Since fruit fly neurons have characteristic arbors, it's possible to obtain information about neuronal shapes through light microscopy. Janelia's Fly Light project [2] visualizes neurons in the *Drosophilia* nervous systems.

The Janelia FlyEM dataset contains the partial connectome from seven columns in the fly medulla [5], with a majority of the information from the fly's adult visual system, which has been extensively studied in terms of its different neuronal types. The fly's visual connectome contains many different neuronal circuits, corresponding to different functions in the brain. Traditional data analyses of the connectome have focused on using topological data analysis features to find circuits and other features of interest within the connectome graph.

Janelia's reconstruction of the seven column dataset is by far the most comprehensive in the field. With this large dataset, we hope to uncover more information about circuit dynamics from structural connectome information. For example, Takemura et. al. attempts to learn about the motion detection circuit while constructing the seven column dataset [10].

Recent advancements in deep learning have shown promise in predicting or classifying data given structural representations. These deep learning techniques apply naturally to the connectome, as we can visualize the fruit fly connectome data in a graphical format. Recently, work has been done to apply neural networks to arbitrarily structured graphs. We hope to apply graph convolutional

networks to the fly connectome data to infer functional information of neurons from given structural information. Specifically, given structural data relating to the location of a neuron, along with graphical information such as the number of neighbors that neuron synapses with, we hope to predict aspects of the neuron's function. In the fly adult visual system, this may take the form of predicting whether a cell is a photoreceptor or not.

We formulate this problem as a semisupervised learning problem to provide a useful tool for neuroscientist in the field. While neuroscientists often have labeled specific neurons in the brain, but labeling every neuron in the brain would be extremely time consuming. Using deep learning, we can use structural/functional information from a few human labeled neurons to predict functional information from surrounding neurons, without a neuroscientist having to label those neurons. The predicted functional information can be used as a guide for more detailed labeling of neurons, guiding neuroscientists towards relevant neuronal circuits in less time.

2 Task definition

The task breaks down into two categories:

(1) Organizing the Janelia FlyEM data into the format of a structured graph. This means we need to organize the data into some graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that $\mathcal{V}$ is the set of all vertices or neurons, and $\mathcal{E}$ is the set of all edges or synapses.

(2) Given features for each neuron and a set of functional classification labels, output the class for each neuron, as labeled by the Janelia team. In many cases, class corresponds to some specific function.

3 Background and related work

For our project, we wish to analyze neurons and synapses from a single column the home column. We can represent this data as a graph and apply computational methods to learn features from this graph. In the past, scientists have applied topological models to analyze column structure. Masulli and Villa [9] use algebraic topology techniques to encode link structures of edges in directed graphs, specifically in neuronal models, via the directed clique (symplitial) complex. In this model, nodes represent neurons, and edges represent synapses (connections) between neurons with a given strength or weight. Masulli and Villa successfully compute the Euler characteristic for all the nodes in the network and each node active during the evolution of subnetworks. The Euler characteristic is observed to explain pruning activity during neural network evolution specifically, an increase in the Euler characteristic corresponds to the number of active neurons in a subnetwork increasing and vice versa.

Related work has also been done in applying traditional machine learning techniques to connectome data. Brown and Hamarneh [3] compare several different models for human brain fMRI and MRI data, discussing different feature schemes. Linear prediction models and SVMs were most commonly used to model feature distributions and improve dimensionality reduction techniques, extracting useful or salient information from complex brain networks. A table of their results can be found online [1]. This paper argues that deep learning will become more feasible as the size of our connectome datasets increases, and models can be trained well enough to capture the large variability found in the brain.

Applying well-established models like RNNs and CNNs to work on these graphs is generally difficult. Kiph and Welling [7] propose the framework of graph convolutional networks (GCNs). The goal of GCNs is to learn some features of a graph given a table of features for each node and adjacency matrix of edges. We apply the semi-supervised learning algorithm described in the paper to our connectome data for a single column, which in theory will help separate our neurons into different classes. It is worth noting that Kiph and Welling have also applied GCNs to Variational Graph Autoencoders [8], but the task of link prediction on simplitial complexes (complete graphs) is less appealing than that of neuronal classification, as a useful tool for neuroscientists in the field.

4 Approach

Kipf and Welling have focused on semi-supervised classification of the nodes in an input graph, which requires that a few nodes in each possible output class be labeled. Since the fly visual system has been well studied, it is indeed possible to start with the labels for a few types of neurons to provide a starting point. Indeed, even in datasets that are less well studied, it is usually possible to start with the labels for a few types of known neurons, or at least with their broad categories.

Using a semi-supervised approach here is ideal because it is hard to obtain large quantities of precise neuronal data using EM techniques, and we would like to use a small amount functional data about neurons and the structural graph of the connectome (neurons and synapses) to predict functional data across all of our neurons.

We take these subgraphs of the connectome and apply our GCNs to yield classifications and learned features of the graph, which embed the graph in a latent space. We also examine the predicted output classes of the neurons in these subgraphs to see if similarly classified nodes within the subgraph correspond to biologically relevant neural circuits.

4.1 Graph Convolutional Networks

A GCN takes in the following input:

- X: an N by D feature matrix, where D is the number of input features and N is the number of nodes
- A: an N by N adjacency matrix

Given this input, the GCN tries to learn the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and predict outputs or labels y_i. Kiph and Welling propose the following layer-wise propagation rule for each neural network layer:

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

Here, $\tilde{A} = A + \mathbb{I}$ and $\tilde{D}$ is the diagonal node degree matrix of $\tilde{A}$. With spectral analysis, we can define a graph convolution as multiplication of spectra of signals in a Fourier domain. However, computing an exact graph convolution is $O(N^3)$, which is computationally expensive and motivates an approximation for a graph convolution. One way to do this is by expressing the convolution kernel using Chebyshev polynomials of eigenvalues in a spectral domain, as proposed by Defferrard, Bresson, and Vanderghenyst [4].

We apply a softmax activation function row-wise to $f(X, A)$ as defined in equation 1 to obtain $Z = softmax(f(X, A))$, where $softmax(x_i) = \frac{exp(x_i)}{\Sigma_i exp(x_i)}$. To evaluate loss in our semi-supervised model, we calculate cross-entropy error as follows:

$$\mathcal{L} = -\Sigma_{l \in \mathcal{Y}_L} \Sigma_{f=1}^{F} Y_{lf} ln Z_{lf} \tag{2}$$

where $\mathcal{Y}_L$ is the set of nodes with labels, or our labeled training instances. The weights of the neural network W are trained using gradient descent with an Adam optimizer [6].

4.2 Preprocessing

We preprocess Janelia's FlyEM dataset as per the first category of our task definition. Specifically, we preprocess the data into three data structures for our GCN. The Janelia dataset consists of 462 well characterized neurons from 14 different classes.

Let $N = 462$ be the number of neurons or vertices, $D = 47$ be the number of features per neuron, and $E = 14$ be the number of binary classes or output labels for each neuron. The data structures include:

1. N by $N(0, 1)$ adjacency matrix representing synapses between neurons
2. N by D feature matrix

3. N by E label matrix

Each feature is one-hot encoded, and the labels or classes are binary (1 is being in the class; 0 is not being in the class).

In Janelia's dataset, we have access to 462 neurons or vertices. Each neuron has a unique ID and features relating to the neuron's location and function. Descriptions of the feature categories are provided below.

$Feature$	Description
Class	Output category. Different cell types are grouped by structure and function. Examples include the Ttype neuron 4 (T4) neuron class, which is involved in the motion detection circuit.
Superclass	Different cell classes are grouped into a superclass by common arborization patterns. Examples include Photoreceptors, Proximal Medulla Neurons, and Y Cells.
Columnar Spread	Takes the value of "Single Columnar" or "Multicolumnar" depending on the percentage of the neuron's arborization that lies in a particular column. If > 0.9 of the cell's synapses lie in a particular column, that cell is labeled single columnar and its column value is recorded.
Columnar Location	Takes the value of "Interior" or "Exterior" based on whether a majority > 0.5 of the neuron's arborization is within the seven columns of the medulla.
Column ID	This field is only defined for single columnar, interior neurons; it denotes which of the seven medullar columns the neuron lies in.
Column Volume	The specific fraction of the neuron's total volume that lies within the seven columns. Defined for every reconstructed neuron.
Synaptic Fractions	This feature has four categories: what percentage of the neuron's dendrites and axons lie in each specific column, and what percentage lie in each of the ten layers of the medulla.

The edges of the graphs are defined by the synapses file, where each synapse is defined by its X, Y, Z coordinates and the ids of the two neurons it connects. We only consider the synapses (edges) between these 462 neurons.

4.3 Classification

We use a hybrid Chebyshev GCN and Multi-layer perceptron architecture, as described in section 4.1 to classify each neuron in our test set. Our model includes a Chebyshev graph convolutional layer and two dense hidden layers. We approximate with a Chebyshev polynomial of degree 1, analagously to Kipf and Welling, to help avoid overfitting.

In addition, the model includes dropout in the Graph Convolutional Layer and the second dense layer to help avoid overfitting.

We split the data as follows: 80% to the train/validation dataset and 20% to the test set. Within the training/validation data, we use only 25% of the data for labeled training data, and the remaining 55% as unlabeled validation data, as per the semisupervised learning approach.

We then apply our GCN to predict the class of the neuron.

5 Experiments

Our model includes a Chebyshev graph convolutional layer and two dense hidden layers. This classifier model has five hyperparameters: the number of hidden units,h; the number of epochs, e; the learning rate l, and the dropout $p = 0.5$.

We performed hyperparameter evaluation by systematically varying these parameters. Several of the determination tables (showing accuracies) are shown below.

After finding the optimal model and number of hidden neurons, we moved on to optimize the learning rate of the model. The optimal learning rate with 1000 epochs was 0.10.

l	Accuracy
0.001	59.140
0.01	65.091
0.10	**70.968**

Figure 1: Variation of l with other hyperparameters fixed at $h = 128$, $e = 1000$, and $p = 0.5$, with test accuracy shown in the second column

$Epochs$	Accuracy
500	53.763
750	64.516
1000	**74.194**
1250	54.839
1500	68.819
1750	53.763

Figure 2: Determination of optimal model with $h = 128$, $p = 0.5$, $l = 0.10$, with test accuracy shown in the second column

$HiddenUnits$	Accuracy
32	44.086
64	52.688
128	74.194
256	49.462

Figure 3: Determination of optimal hidden neuron value h with $p = 0.5$, $e = 1000$, $l = 0.10$, with test accuracy shown in the second column

The optimal hyperparameters were:

$$h = 128 \qquad e = 1000 \qquad l = 0.1 \qquad p = 0.5 \tag{3}$$

A final run on the test set resulted in a test accuracy of 74.195%. This accuracy is significantly higher than random for a 14-class classification problem. In addition, note that this accuracy is higher than the average validation accuracy, for reasons that are not well understood.

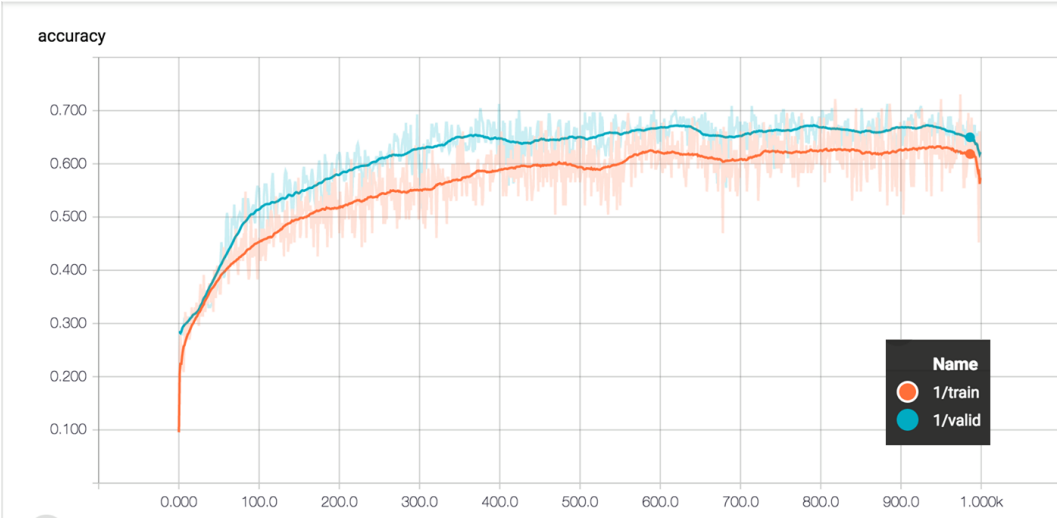

Figure 4: Training and validation set accuracies

6 Future Research

There are a number of interesting extensions that can be performed on this research. The first extension is to supplement our model with additional structural data, and further examine which structural features are correlated with classification into certain neuronal classes. In addition, we also wish to examine which classes of neurons this model performed best and worst on.

On the biological side, we would like to extend this model to new datasets and cases where this semisupervised learning approach could be valuable to research in labeling populations of neurons.

Acknowledgments

We would like to thank Tom Dean and Amy Christensen for their guidance in this project. We would also like to thank Janelia for access to fly connectome data, and Google Cloud for access to computing resources.

References

[1] Connectome Learning Table.

[2] FlyLight.

[3] C. J. Brown and G. Hamarneh. Machine Learning on Human Connectome Data from MRI. *ArXiv e-prints*, Nov. 2016.

[4] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016.

[5] janelia flyem. Janelia ConnectomeHackathon 2015.

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[7] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[8] T. N. Kipf and M. Welling. Variational Graph Auto-Encoders. *ArXiv e-prints*, Nov. 2016.

[9] P. Masulli and A. E. P. Villa. The topology of the directed clique complex as a network invariant. *ArXiv e-prints*, Oct. 2015.

[10] S.-y. Takemura, A. Bharioke, Z. Lu, A. Nern, S. Vitaladevuni, P. K. Rivlin, W. T. Katz, D. J. Olbris, S. M. Plaza, P. Winston, T. Zhao, J. A. Horne, R. D. Fetter, S. Takemura, K. Blazek, L.-A. Chang, O. Ogundeyi, M. A. Saunders, V. Shapiro, C. Sigmund, G. M. Rubin, L. K. Scheffer, I. A. Meinertzhagen, and D. B. Chklovskii. A visual motion detection circuit suggested by drosophila connectomics. *Nature*, 500(7461):175–181, Aug 2013. Article.