Constructing functional graphs and recurrent neural network models of neural dynamics using calcium imaging data in Python

Veronica Porubsky

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:

Herbert M. Sauro, Chair

Michael R. Bruchas

Joseph L. Hellerstein

Program Authorized to Offer Degree:

Bioengineering

University of Washington

Abstract

Constructing functional graphs and recurrent neural network models of neural dynamics using calcium imaging data in Python

Veronica Lynn Porubsky

Chair of the Supervisory Committee:

Herbert M. Sauro

Department of Bioengineering

Leveraging the power of computational analyses and modeling could assist with the study of neural dynamics. Complex analyses like graph theory could enable biomarker discovery in psychiatric illnesses, and recurrent neural network models could be used to reveal mechanisms underlying behavior and cognition. More broadly, both graph theory and recurrent neural networks could be used to understand the topology of functional networks in microcircuits within the brain. This research aims to generate software tools that neuroscientists can readily use to assist experimental analyses of neural data while remaining extensible for more complex analyses. Two Python packages being developed towards this goal are shared: cagraph and carnn. These packages support graph theory analysis and recurrent neural network modeling of neural systems that have been recorded using calcium imaging.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I want to acknowledge my advisor, Dr. Herbert Sauro, who set me on the path to becoming a computational researcher and provided several opportunities to improve my leadership and science communication skills as the Head of Outreach for the Center for Reproducible Biomedical Modeling. He introduced me to many incredible computational biologists worldwide who comprise the COMBINE community, and together they showed me the power of collaboration and community in science. In this role working with the Center for Reproducible Biomedical Modeling, I was generously supported by the National Institute of Biomedical Imaging and Bioengineering of the National Institutes of Health under award number P41-EB023912.

I want to thank my collaborators, Dr. Sean Piantadosi and Eric Zhang, members of the Bruchas Lab at the University of Washington, who have graciously shared their datasets with me for my computational studies and provided great insight into the experimental methods. They helped guide my research through fruitful discussions and have offered me extensive feedback throughout my Ph.D. It has been an honor to work alongside them. I want to acknowledge the members of the Sauro Lab, especially Dr. Lucian Smith, and the remaining members of the Bruchas Lab, who provided feedback on my work through regular lab meeting presentations and informal discussions. I want to acknowledge the UW Center of Excellence in Neurobiology of Addiction, Pain, and Emotion (NAPE) for providing training in the experimental details of calcium imaging data collection and experimentation through the 2022 Calcium Imaging Workshop. I also want to thank Dr. Charles Zhou and the NAPE Computing Subgroup for additional discussions on the computational details of my research.

I want to acknowledge the members of my Ph.D. Supervisory Committee - Professor Herbert Sauro, Professor Amy Orsborn, Professor Michael Bruchas, Professor Joseph Hellerstein, and Professor Larry Zweifel - for advising me on my Ph.D. progress and providing feedback and support throughout my graduate studies. I was incredibly fortunate to work with these individuals, who are not only incredible researchers but also willing to mentor and challenge their students, encouraging them to succeed. Professor Bruchas and Professor Orsborn welcomed me into their lab meetings at different stages of my Ph.D. I am immensely grateful to have met such incredible scientists and kind, helpful humans in their labs, in addition to learning from their leadership and technical expertise. Professor Joseph Hellerstein offered substantial mentorship to me, meeting with me to discuss the computational details of my research and plans for my career. He has consistently demonstrated himself to be an advocate for me and many of my colleagues in the Sauro Lab.

I am incredibly grateful to my friends and family who have supported me throughout my life, especially during the time I spent at the University of Washington pursuing my Ph.D. They have always been there for me: to listen and to be leaned upon in times of need. My parents - James and Andrea Porubsky - have always encouraged me to persist despite obstacles and to work hard. My brothers - Dr. Nicholas Porubsky and Dominic Porubsky - are both incredible role models to me, and never hesitate to give their advice and encouragement. I will always cherish the relationships I have with these dear humans and hope to always return the support to them.

DEDICATION

To my family and to the friends I consider family.

Chapter 1. INTRODUCTION

Studying neural circuits provides essential information about the mechanisms the brain uses to process information and produce complex behavior. With recent improvements in the experimental methods used to collect large-scale and high-resolution neural activity in live and behaving systems, there is a need for effective methods to interpret the data using both trial-averaged data single-trial data. Due to the complex nature of the data itself, neuroscientists require advanced computational analysis methods. Traditionally, this would only be possible through collaboration with specialists familiar with the statistical and programmatic methods used to develop advanced computational tools. However, the work described in this thesis aims to make two of these methods readily accessible to neuroscience research using cellular resolution calcium imaging data in their research.

The key objectives of the work described are to make easy-to-use and robust software tools to support studying neural dynamics using graph theory and recurrent neural network models. Here, the robustness of the software refers to the validation of the applied mathematical approaches when used on calcium imaging time series data. For example, the software should be tested to ensure it can accommodate calcium imaging data that has undergone distinct preprocessing steps to extract the time series. The software should be useable for experimental neuroscientists who collect calcium imaging data, and should still be accessible to researchers with minimal programming experience. The work demonstrates the utility of these studies by applying them to analyze neural data, which has been collected from multiple regions of the brain under several behavioral paradigms in mice. The work described in the following pages makes a significant and original contribution to the scientific community by providing two

Python packages to augment the study of neural circuits. These packages have been applied to calcium imaging data and used to demonstrate that functional connectivity patterns change in response to behavioral stimuli in the context of a fear conditioning paradigm.

The remaining chapters will provide a literature review showing the importance of analyzing neural circuits to understand how information is integrated to give rise to complex behaviors. Two research chapters outline Python packages being developed to analyze calcium imaging data, a third research chapter shows the application of these packages to a case study in the context of fear processing, and a conclusion section summarizes the results of these research chapters and future directions for the research described. The first research chapter will provide an overview of the cagraph Python package. This package converts calcium imaging data into a graph object for analysis of the functional connectivity of the recorded neural data. The second research chapter will provide an overview of the carnn Python package. This package is under development and designed to convert calcium imaging data into recurrent neural network models capable of recapitulating the neural dynamics of the original system. The third and final research chapter will demonstrate the functionality of cagraph and show that graph theory can be used to identify changes in functional connectivity of the anxiety circuitry.

This thesis will demonstrate the functionality of these two software tools, which support the analysis of neural data collected using one-photon and two-photon calcium imaging, and show the audience the ease of using these tools. The proposed methodology and associated Python packages offer functionality to readily convert singular and batched calcium imaging datasets into objects which can be used to perform graph theory analyses or to construct recurrent neural networks (RNNs) to extract meaningful conclusions about the data or to perform experimentation on the model of the system *in silico*. The work provides a framework for

analyzing calcium imaging data with graph theory and generating recurrent neural network models of neural circuits. The proposed approach is highly flexible and can be applied to various behavioral settings, including in vitro and in vivo studies and experimental studies which collect calcium imaging data. The software tools developed in this thesis can be used by neuroscientists to gain insights into the complex dynamics of neural circuits and to test hypotheses about their function.

1.1 AIMS FOR THE DISSERTATION WORK

Neuroscience data has become increasingly easy to collect at finer resolution, and experimentalists can readily collect high-dimensional datasets comprising networks of thousands of neurons at cellular resolution. However, analysis methods capable of deciphering the patterns in these data are constantly being developed and adapted from other disciplines to understand neural data. The complex theory behind these methods and the tools available to support them can preclude non-specialists from using them without substantial training. This challenge necessitates the development of new tools which are accessible to non-specialists.

The following aims are addressed in the upcoming chapters of this thesis. Additional proposed analyses and development are described when relevant. Aim 1 is addressed in Chapter 2 and Chapter 3, and the remaining work required for Aim 2 is discussed in Chapter 4.

Aim 1. Develop and validate a Python package to study functional connectivity patterns using graph theory.

Aim 1.1: Combine the functionality of existing Python packages for graph theory analysis and network visualization to construct functional graphs of calcium imaging datasets.

Aim 1.2: Analyze calcium imaging datasets collected from the dentate gyrus and basolateral amygdala in mouse models of anxiety to demonstrate the utility of the resulting package.

Aim 2. Design a Python package to build recurrent neural network models that replicate neural dynamics for experimentation *in silico*.

Aim 2.1: Provide the design for an accessible software package containing construction, training, and simulation utilities of recurrent neural network models capable of representing calcium imaging data using powerful libraries for deep-learning.

Chapter 2. A PYTHON PACKAGE FOR CONSTRUCTING FUNCTIONAL GRAPHS OF CALCIUM IMAGING DATA

Neural systems can be studied using network neuroscience, built upon graph theory, to unravel the structural and functional connectivity that gives rise to cognition and behavior – both healthy and aberrant. While large-scale structural connectivity has been a primary focus of these studies to date, more recently, inquiries about how networks are functionally connected – or correlated in time – have led to a deeper investigation into the topology of network dynamics. Network topology interests neuroscientists working to understand neural dynamics within and across brain regions. Calcium fluorescence imaging is commonly used to collect neural dynamics of a

population of individual neurons in live and behaving systems to characterize functional microcircuits within a single region. However, there is no existing Python package for neuroscientists to use that can readily construct graphs based on calcium imaging data and perform graph theory analyses. Here, a Python package called cagraph is described, and its functionality to readily perform graph construction and analysis is demonstrated. Software usage examples are shown. This package is open-source and available for installation via the pip package installer.

2.1 BACKGROUND

The brain processes information on multiple spatial and temporal scales – at individual synapses, through local microcircuits within a single brain region, and distributed interactions across distinct brain areas. Network neuroscience has been leveraged to discover the mechanisms by which structural and functional connectivity create networks in the brain that give rise to cognitive and behavioral processes [1]. This field of inquiry draws from the discrete mathematical approach called graph theory [2], which represents members of a system (nodes) and the interactions between them (edges) in a graph. This graph can be used to study the complex patterns of interactions, also called the network topology, that arise in the system. Graphs can be used to study many distinct system representations, including spatial and temporal data relating entities in the system and covering sub-cellular, cellular, and structural networks encompassing the range of neural computation.

Of these numerous mathematical methods that can be used to analyze neural activity, graph theory is commonly applied to dynamic time series data to identify functional connectivity based on the correlation structure of electrophysiological events that arise in the neural data.

Graph theory can be used to dissect the synchronicity of neural ensembles and understand network topology when studying neurological systems [3]–[12]. Graph theory involves the study of graphs, which are mathematical structures that model the relationships between objects and are comprised of a set of nodes and edges connecting them. Neural time series can be converted into graph structures by representing each neuron as a node in the graph, and the correlation between two time series can be used to represent the edges.

Graph theory has many methods which can be used to show patterns in the functional connectivity of neural systems. For example, communities, which are densely connected clusters that have sparse connections to other clusters, have been shown to correspond to functional modules in some systems within the brain [10], [11]. Hubs, which are nodes that have connections to many other nodes in the network, have also been shown to be crucial to information processing in the brain [10], [11]. Overall, by applying graph theory to the study of functional networks, researchers can gauge the synchronization of neural activity, which follows from the tendency of neurons to fire together in a coordinated manner to process a stimulus or simply as a response to upstream neurons that have synapses onto several of these neurons in the recorded network and are therefore driving the response.

By studying a neural population over multiple days or in response to different behavioral stimuli, it is also possible to assess the stability of the network by determining whether functional connections are maintained in response to external perturbations. While graph theory is only one method that should be considered in the analysis of neural dynamics, it can serve to study complex activity patterns with a well-established interpretation of the analyses.

Using network neuroscience approaches, researchers have managed to generate maps describing the structure and function of networks in the human brain [13] and such studies have

shown that the topology of these brain networks is typically conserved across species in terms of the large-scale connectivity between regions [14], [15]. Beyond these healthy networks, it has been suggested that alterations in the structural and functional connectivity of the brain can give rise to cognitive and behavioral illnesses. These networks, too, have been investigated to understand how the changes in neural activity can interrupt healthy functioning [16]–[18]. However, with many studies focusing on structural differences, it has been challenging to translate this understanding into models that represent the underlying dysfunctional network.

While there have been some attempts to construct MATLAB and Python packages to perform graph theory analyses on calcium imaging data, these have been mainly constrained to wide-field calcium imaging [19], which gives a dorsal whole-brain analysis of transgenic animal brains. In addition to providing the graph theory analysis of neural dynamics, the cagraph package provides methods to integrate experiment-specific metadata and behavioral data, which can assist with sampling discrete portions of the network or the neural time series used to create the graphs.

2.1.1 *Introduction to graphs*

A graph is a collection of nodes (or vertices) connected through edges. The edges can be directed, meaning they show a direction from one node to another, or undirected, meaning each edge shows a relationship between two nodes without an associated directionality. Additionally, simple graphs contain no self-loops (edges between a node and itself) or parallel edges (repeated edges between the same two nodes).

The next distinction is between weighted and unweighted graphs. Edges in weighted graphs contain numerical weights indicating the strength of the correlation between the two connected nodes. In an unweighted graph, there is no weight associated with the edge. Instead,

edges will exist in the graph if the correlation between two nodes is above some positive threshold, or in the case of anticorrelated relationships if the correlation is below some negative threshold value. In the case of functional graphs constructed using calcium imaging data, a node represents the fluorescence data time series collected from a segmented neuron, and the edge either represents the strength of the correlation between two neural time series (weighted graph) or the existence of a correlation that surpasses the threshold value (unweighted graph).

2.1.2 *Calcium imaging and graphs*

Calcium imaging can be used to simultaneously record hundreds of neurons in vitro and in live and behaving animals. The preprocessing of these data can require several steps to extract the location of neurons and record and denoise the activity in the field of view. These processes have been extensively studied and optimized [19]. One method using constrained non-negative matrix factorization (CNMF) for endoscopic data, or one- and two-photon fluorescence microscopy data, has been described, which effectively demixes overlapping neural signals and denoises the data to extract neural time series [20].

While networks of neurons have directed relationships where temporal signals of activation in one neuron lead to the activation of other postsynaptic neurons, the temporal resolution of calcium imaging recordings can be too low to extract these directed relationships. Even in cases where the resolution is sufficiently high, the field of view for micro-endoscope data may not capture subpopulations of neurons that terminate on the same recorded population. As a result, it is reasonable to construct functional connectivity graphs using Pearson's correlation coefficient to establish relationships in undirected graphs.

2.1.3 *cagraph: a Python package for graph theory analysis of calcium imaging data*

Here, an open-source Python package called cagraph is described, designed to analyze calcium imaging data collected with one- and two-photon calcium imaging using graph theory with case study examples demonstrating the utilities of the package. The applicability of the package to a one-photon dataset is shown explicitly. The package simplifies the construction of functional graphs that represent the dynamics of local neural microcircuits and executes graph theory analyses on these graphs to understand the topology of the network. cagraph provides preprocessing and visualization functionality to readily inspect the dataset and graph. Beyond simplifying the creation of functional graphs of calcium imaging data, it offers shared functionality that has been robustly validated for neuroscientists to check that their dataset is suitable for this graph theory analysis and provides functionality to visualize and analyze their data. cagraph has been distributed as a pip-installable package, simplifying the installation process to one line of code. The package is maintained on the Python Package Index (PyPi) server as a stable software version, tested before deployment. cagraph v1.0.0 will support the analysis of simple, unweighted, and undirected graphs. Support for weighted and directed graphs is in progress and expected to be included in the next version of the software.

In addition to a simple analysis of singular datasets, additional classes have been created within the cagraph package to provide utilities for batched analyses, analyses that split single datasets into distinct periods sampled from the time series, analyses of matched or tracked datasets that have a subset of cells that are the same in each dataset, and analyses of behavior-sampled datasets where the time series is sampled according to the behavior happening over the recording session. By adding these analyses, this package provides the opportunity to analyze

functional graphs of calcium imaging data and incorporates critical information about the behavioral experiments under which the neural dynamics were recorded.

Packages for applying graph theory to calcium imaging data have been implemented previously. In the case of mesoscale calcium imaging data, O'Connor et al. constructed a pipeline for analyzing wide-field calcium imaging in Python that provides similar graph theory analysis functionality and uses many of the same dependencies as the cagraph package [21]. Previous work has applied graph theory to voltage-gated mesoscale imaging data [22] using a MATLAB toolbox called the brain connectivity toolbox [23], which is similar to NetworkX while providing additional analyses geared towards neuroscience and brain research. While the MATLAB toolbox supports complex network analysis of neural data, it is not open-source and therefore limits access. The cagraph package is constructed to be readily installable and open-source for all to use. While the pipeline described by O'Connor et al. is quite similar, it is optimized for mesoscale calcium imaging. It also provides different visualization and plotting functionality and does not provide utilities for parsing graphs using behavioral information, as is available in the cagraph package.

2.2 METHODS

The cagraph package is built using several open-source Python packages which support mathematical and scientific computing in Python. The basic numerical computations are performed using Numpy [24], and scientific computing, including statistical analysis, is achieved using SciPy [25]. The graph construction and analysis are performed using NetworkX [26]. The underlying Bokeh library allows users to visualize the graphs generated in cagraph, offering interactive visualization output that can be inspected with greater detail and metadata than the

static drawings in NetworkX [27]. For plotting and statistical visualizations, Matplotlib [28]and Seaborn [29] are used. The reports generated using the software are handled using the Pandas package [30].

2.2.1 *Preprocessing pipeline for one- and two-photon calcium imaging data*

The preprocessing steps required for extracting calcium fluorescence data for the cagraph package are executed using a standard pipeline. Constrained non-negative matrix factorization for endoscopic data (CNMF-e) is recommended for extracting neural traces. Constrained non-negative matrix factorization for endoscopic data (CNMF-e) [20] was used to extract individual neuronal calcium fluorescence traces for the datasets used to test the cagraph package. The package is best suited for CNMF-e extracted traces.

The preprocessing module provides the OASIS method for deconvolution of calcium imaging data, which can be helpful for datasets that are not extracted using CNMF-e [31], [32]. For datasets that have not been deconvolved prior to using the cagraph package, the data should first be processed using the deconvolution functionality.

```
import preprocess as prep
deconvolved_data, event_data = prep.deconvolve_dataset('dataset.csv')
```

2.2.2 *Input data types to the CaGraph class*

Currently, the CaGraph class supports input data as loaded numpy.ndarray types, CSV files, and simple Neurodata Without Borders (NWB) files consist of a single HDF5 file. The user must specify the full path to the CSV or NWB file if the dataset is not loaded into memory as a numpy.ndarray. The input data must be oriented such that the first row of the data represents the time points at which the calcium fluorescence imaging session was sampled. The subsequent

rows must include the calcium fluorescence traces for each neuron in the network, sampled at each time point specified in the time row. Then, each column represents the state of the recorded neural population at a single time point.

	0	1	2	3	4	...	4996	4997	4998	4999	5000
time	0.100000	0.2000	0.300000	0.400000	0.500000	...	4.997000e+02	4.998000e+02	4.999000e+02	5.000000e+02	5.001000e+02
neuron 0	2.331200	2.2932	2.255900	2.219100	2.183000	...	1.416100e+01	1.393100e+01	1.957100e+01	1.925200e+01	1.893900e+01
neuron 1	5.134400	5.0265	4.920800	4.817400	4.716100	...	5.202400e+00	5.093100e+00	1.083800e+01	1.758600e+01	1.721600e+01
neuron 2	0.000000	0.0000	0.000000	0.000000	0.000000	...	2.592200e+01	2.579300e+01	2.566400e+01	2.553600e+01	2.540900e+01
neuron 3	2.732700	2.7124	2.692200	2.672200	2.652300	...	5.501500e+00	5.460600e+00	5.420000e+00	5.379700e+00	5.339600e+00
...	...	...	...	...	...	...	...	...	...	...	...
neuron 34	2.010200	1.9488	1.889300	1.831600	1.775700	...	1.004300e-24	9.736300e-25	9.439000e-25	9.150900e-25	8.871500e-25
neuron 35	0.000000	0.0000	0.000000	0.000000	0.000000	...	1.081100e-141	9.846900e-142	8.968800e-142	8.169000e-142	7.440500e-142
neuron 36	0.009955	0.0097	0.009451	0.009209	0.008972	...	8.950900e-24	8.721300e-24	8.497600e-24	8.279700e-24	8.067300e-24
neuron 37	0.000000	0.0000	0.000000	0.000000	0.000000	...	3.091800e-09	3.052600e-09	3.014000e-09	2.975900e-09	2.938200e-09
neuron 38	0.000000	0.0000	0.000000	0.000000	0.000000	...	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00

Figure 1: Input data format.

2.2.3 *Computing correlations and constructing the interaction matrix*

The Pearson's correlation matrix can be computed from all pairwise combinations of neurons recorded in the field of view. The Pearson's correlation matrix defines all interactions between neurons in the network. All neurons are represented in the graph as nodes, and neuron pairs with a Pearson's correlation coefficient above a predefined threshold value will be added to the corresponding graph with an edge connecting them.

2.2.4 *Establishing thresholds with shuffled neural time series*

Using a shuffled distribution of neural data, the user determines a threshold for setting edges in the graphs. The user will perform hypothesis testing using the shuffled distribution. In the case of the event shuffle, correlations of the calcium events in the dataset are compared to correlations due to randomly shuffled events to test the hypothesis that the correlations between neurons in

the dataset are not statistically different than correlations of shuffled neurons with randomly timed events. If the null hypothesis is accepted, the timing of events is considered to not contribute to meaningful correlations.

Shuffling neural time series using event identification

The event shuffle is performed by identifying distinct calcium transient events for each fluorescence trace from a user-specified event trace, and the calcium fluorescence traces are segmented at each event. The user can supply an existing event trace to the threshold generation algorithm if they have performed the relevant preprocessing steps. If the user does not provide an event trace, this trace will be inferred using the OASIS algorithm for deconvolution of calcium imaging data [31], [32]. All segments are randomly shuffled within the trace.

```
shuffled_data = prep.generate_event_shuffle(data='dataset.csv')
```

Setting a threshold using the ground truth and shuffled correlation distributions

The Pearson's correlation matrix for the shuffled dataset is constructed and compared to the correlation matrix for the ground truth data. A Pearson's correlation coefficient threshold value is selected by computing the 99th percentile of the shuffled distribution. This strict cutoff sets the threshold, above which the ground truth correlations are differentiable from the shuffle dataset. This value can optionally be averaged across mice to establish a shared threshold for the analysis in cases where batches or multiple time samples are processed.

```
threshold = prep.generate_threshold(data='dataset.csv')
average_threshold = prep.generate_average_threshold(data='dataset.csv')
```

2.2.5 *Constructing CaGraph objects*

Calcium imaging fluorescence data is used to construct a CaGraph object for each mouse, defined by correlations computed using Pearson's Correlation coefficient between all pair-wise combinations of neuronal activity time series (nodes). These coefficients are used to assign weights to interactions (edges) in a graph object constructed using the prolific Python package, NetworkX. Graphs can be mathematically represented as G = (V, E). Here, V is the set of nodes in the graph, and E is the set of edges [33]. For undirected graphs, each element in E is an unordered pair {u, v} containing unique nodes u and v ∈ V. The Pearson's correlation coefficient matrix is mathematically computed as follows [34]:

$$R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii}C_{ij}}}$$

Here, the product-moment correlation coefficients are stored in the matrix, R, and the covariance matrix, C, is used to compute the product-moment correlation coefficients described by the equation. The values of R fall between -1 and 1.

By sampling from the whole time series, weight matrices investigating only a subset of the temporal data can be used to assess connectivity, such that the network model will evolve in time, establishing and breaking connections. The graph is constructed by passing a matrix of Pearson's correlation coefficients defining the interactions between all nodes in the network to a NetworkX constructor.

Programmatically, this is accomplished by passing either a numpy.ndarray type object containing calcium fluorescence data loaded into memory as in the following example code:

```
from cagraph import CaGraph
import numpy as np

DATA = np.loadtxt('dataset.csv', delimiter=',')
cagraph_obj = CaGraph(DATA)
```

Alternatively, a path to the dataset to be used can be directly passed to the CaGraph object constructor as a CSV file or a NWB file as in the following example:

```
from cagraph import CaGraph

cagraph_obj = CaGraph('dataset.csv')
```

Analyzing functional connectivity with graph theory

Once graphs are constructed, they can be used for graph theory analyses. This report uses the clustering coefficient to assess the conditions presented (behaviors, contexts, stimulation conditions) in the Chapter 3 case study of the anxiety circuitry. However, the cagraph package currently supports additional analyses, including the identification of hub nodes, the degree of centrality, and the degree of each node. More methods for analyzing functional connectivity patterns will be added in future software updates using the algorithms available in NetworkX.

Using the CaGraph object generated in the construction methods section, essential attributes can be accessed, including the threshold used to assign edges in the graph, a matrix demonstrating the Pearson's correlation between all neuron pairs, the adjacency matrix describing the interactions in the graph, and the networkx.Graph object itself. Providing the user access to the

```
threshold = cagraph_obj.threshold
pearsons_correlation_mat = cagraph_obj.get_pearsons_correlation_matrix()
adjacency_mat = cagraph_obj.get_adjacency_matrix()
graph_obj = cagraph_obj.graph
```

networkx.Graph object ensures that this object can be used with the NetworkX package to perform analyses not currently supported in cagraph.

The graph theory analyses can be accessed in the following manner:

```
cagraph_obj.graph_theory.get_clustering_coefficient()
cagraph_obj.graph_theory.get_hubs()
cagraph_obj.graph_theory.get_communities()
cagraph_obj.graph_theory.get_density()
```

The package is also extensible for advanced users familiar with NetworkX, as methods that are not implemented within the cagraph package directly but which NetworkX offers support for can still be used by using the networkx.Graph object within the CaGraph object. This is shown in the example below using the shortest path length algorithm in NetworkX. Only a subset of the graph theory analysis functionality available through NetworkX is directly supported in the cagraph package. The analyses most relevant to studying the topology of functional microcircuits will be added in updates to the software. By integrating these analyses within the package, they can be added to the output reports and the interactive visualizations without requiring the user to perform the analysis with NetworkX and add the node metadata at the time of object construction to include it.

```
import networkx as nx
graph_obj = cagraph_obj.graph
nx.shortest_path_length(G=graph_obj)
```

Some of the analyses supported in the cagraph package using the underlying NetworkX library are described below.

Clustering coefficient

The clustering coefficient determines the extent to which the neighbors of a node – all nodes that have an edge with the node of interest – are correlated with each other. Therefore, a high clustering coefficient (maximum value of 1) indicates that the neighbors of a given neuron are highly synchronized with each other. Clustering coefficient values lie between 0 (neighbors are not connected) and 1 (all neighbors are connected). The clustering coefficient for each node can be computed as the ratio of the number of triplets formed between neighbors of a node to the number of possible triplets that can be formed. Triplets are formed when two neighbors of the node of interest are also connected with an edge. The clustering coefficient for undirected and unweighted graphs is mathematically computed as follows [35]:

$$CC_v = \frac{2T(v)}{\deg(v) \times (\deg(v) - 1)}$$

Here, CC_v is the clustering coefficient value for node v, $T(v)$ is the number of triplets formed from node v, and $\deg(v)$ is the degree of node v. The degree of the node is the number of edges the node has with other nodes in the graph.

Betweenness centrality for hub identification

Hubs are an important aspect of graph topology, as they represent nodes in the graph that are highly influential and very connected to the graph. Typically, hub nodes are identified by using a centrality measure, which incorporates information about the degree of the node, the degree of the nodes neighbors and the extent to which the node of interest lies on the shortest paths between all pairs of nodes in the graph. In this package, the betweenness centrality is used to identify hubs. Betweenness centrality is based on shortest paths in the graph, where the shortest path between two nodes in an unweighted graph is the smallest number of edges that

must be traversed between two nodes [36]. The betweenness centrality for each node is the number of shortest paths that pass through the that node [36]. Because the betweenness centrality can be very high for large graphs, it is normalized by dividing each value by the number of node pairs in the graph. For undirected graphs, the number of node pairs are:

$$node\ pairs = \frac{1}{2} \cdot (\deg(v))\big((\deg(v) - 1\big)$$

To determine whether nodes in the graph are classified as hubs, the betweenness centrality is first computed for all nodes in the network. Hubs are then identified from the resulting betweenness centrality distribution by selecting outliers from the distribution using the interquartile range method [37]. In this method, the following equation for outlier detection is used:

$$outlier\ threshold = Q3 + 1.5 \cdot (Q3 - Q1)$$

Here, *Q1* and *Q3* are the first and third quartiles, respectively. All nodes with a betweenness centrality score higher than the outlier threshold are labeled as hubs.

Communities

The computation of communities is performed using the greedy modularity maximization, which uses the Clauset-Newman-Moore greedy modularity maximization to find the partition that gives the largest modularity [38]. The modularity is defined in the NetworkX modularity function can be calculated as follows using the reduced formula specified by Clauset et al [39]:

$$Q = \frac{1}{2m}\sum_{c=1}^{n}\left[\frac{Lc}{m} - \gamma\left(\frac{k_c}{2m}\right)\right]^2$$

Here, the modularity, *Q,* is computed by iterating over all communities *c. Lc* is the number of intra-community edges, *m* is the number of edges, and *kc* is the num of the degrees of the nodes in community *c* and is the resolution parameter. The resolution parameter sets a tradeoff between intra-group and inter-group edges. The algorithm begins by placing each node in a separate community and continuously joins the pairs of communities that lead to the most significant increase in modularity into a single community until the maximum modularity is reached.

2.2.6 *Random graphs for comparison*

In addition to providing methods for hypothesis testing with shuffled distributions, functionality is available to generate randomized graphs for comparison to the functional graphs of neural activity. The method used in NetworkX is adapted from the Maslov and Sneppen algorithm and randomly rewires an existing graph using connection probabilities [40].

2.2.7 *Graph theory report*

A standardized report method is available in the cagraph package so that any singular dataset used to create a CaGraph object can generate a complete overview of the network topology. The report automatically provides a subset of essential graph theory analyses. This report is exported to a CSV file or can be analyzed immediately in Python as a Pandas DataFrame object

```
report = cagraph_obj.get_report()
```

2.2.8 *Sensitivity analysis*

A sensitivity analysis function is included to show the extent to which the graph changes when the threshold value is set to values around the recommended threshold. This sensitivity analysis computes the graph edit distance between the graph constructed using the recommended threshold and the graphs constructed using values surrounding the recommended threshold. The graph edit distance is a similarity measure that computes the minimum cost of the edit path required to transform one graph into an isomorphic second graph it is being compared to [41]. The edit path is the sequence of nodes and edges that must be modified to make the two graphs isomorphic. Therefore, the sensitivity analysis will show low values of the graph edit distance metric for threshold values that are very close to the identified threshold, but this metric will increase as the threshold is moved further from the initial value.

```
cagraph_obj.sensitivity_analysis()
```

2.2.9 *Interactive network visualization*

The Bokeh package generates interactive network visualizations that can be exported as HTML files [27]. These interactive networks show all nodes and edges in the graph by default, and information about the local connectivity and global structure can be included in the visualization using hover tools, where moving the mouse over each node reveals additional data and metadata about that node.

```
import visualization as viz
viz.interactive_network(cagraph_obj=cagraph_obj)
```

2.3 RESULTS

2.3.1 *Preprocessing examples*

The cagraph package requires that time series data for a population of neurons has been extracted from calcium imaging movies. The package has been tested primarily on CNMF-extracted data, indicating that some preprocessing has already been applied to denoise the fluorescence traces. The preprocessing functionality provided by the cagraph package performs a series of checks to ensure that the provided dataset is suitable for the graph theory analysis.

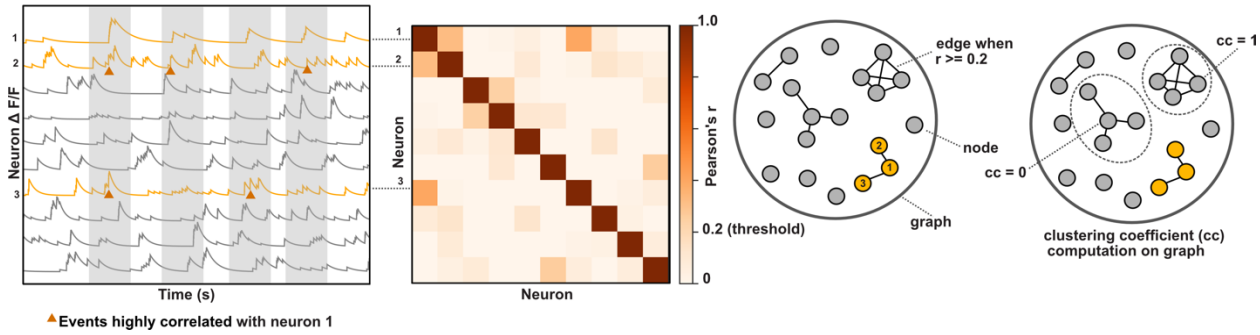

Figure 2: Schematic of graph creation using calcium imaging data.

The preprocessing module provides functionality to set a recommended threshold Pearson's correlation coefficient value, ultimately determining whether edges are added to the graph. The preprocess threshold setting functionality first shuffles the neural activity using event detection. Each calcium event identifies a time to split the time series. Within each neuron, the time series is shuffled. This event identification and shuffling procedure is demonstrated in Figure 3.

Pearson's correlation coefficient is calculated between all pairs of neurons in the dataset and in the shuffled dataset, and the correlation distributions for the data and the shuffle are compared using the Kolmogorov-Smirnov test [42]. This test examines whether two distributions

are statistically different from one another. The hypothesis tested is whether the correlations between observed neural traces are due to meaningful timing of calcium transients or if the exact timing is insignificant and the correlations are the same as the random shuffle.

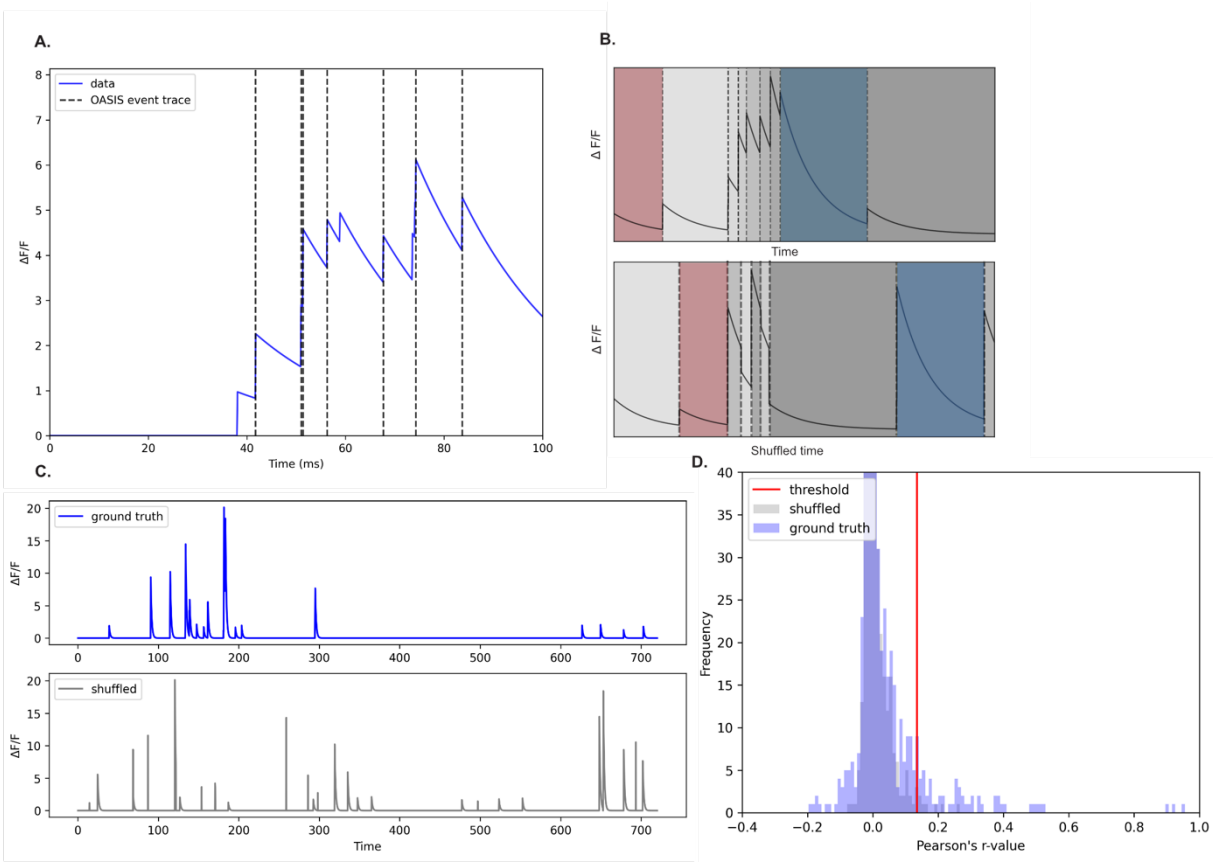

Figure 3: Preprocessing module functionality.

The next preprocessing functionality is a sensitivity analysis which allows the user to explore the effects of setting the threshold to values surrounding the recommended threshold. An example sensitivity analysis is demonstrated in Figure 4. If changing the threshold value by small amounts around the threshold significantly disrupts the functional connectivity observed in the graph, the dataset is sensitive to the threshold. In cases where there is a high degree of variability in response to small changes, it becomes critical to set a threshold that will identify meaningful correlations. Typically, there is a heightened sensitivity to lowering the threshold, as

most of the correlation distribution is below the recommended threshold, suggesting that setting a lower threshold would greatly increase the number of connections in the graph.

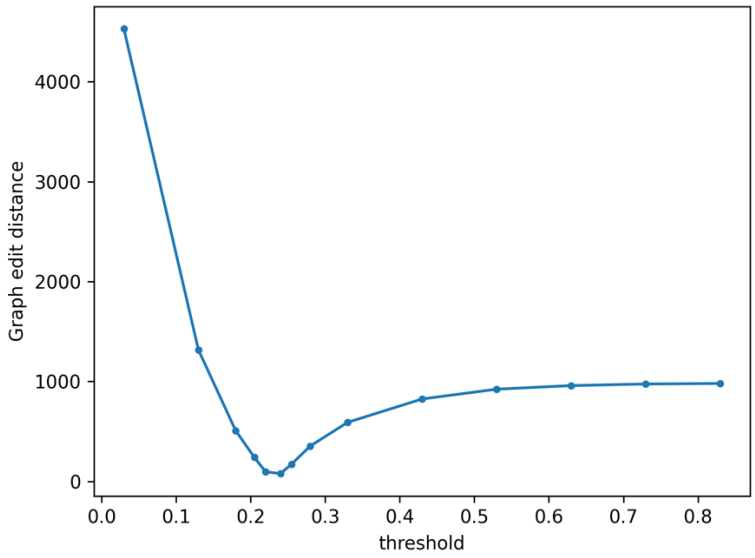

Figure 4: Sensitivity analysis of the threshold value.

2.3.2 *CaGraph construction example*

CaGraph objects form the foundation of the cagraph package, as they are constructed on a single calcium imaging dataset, building a graph that can be readily analyzed and visualized. The CaGraph object constructor minimally requires the calcium imaging dataset loaded as a numpy.ndarray or passed as the path to a CSV or NWB file format to create the graph. Optionally, the user can pass a predetermined threshold value. If the user does not provide a threshold value, a threshold will be automatically generated using the preprocessing module.

Pearson's correlation coefficient between all pairs of neurons is computed to generate a Pearson's correlation matrix. The graph is constructed using the adjacency matrix created by applying the threshold to the Pearson's correlation matrix. The resulting CaGraph object contains

a networkx.Graph object that can be used to compute many graph theory metrics, and therefore used to investigate the topology of functional networks.

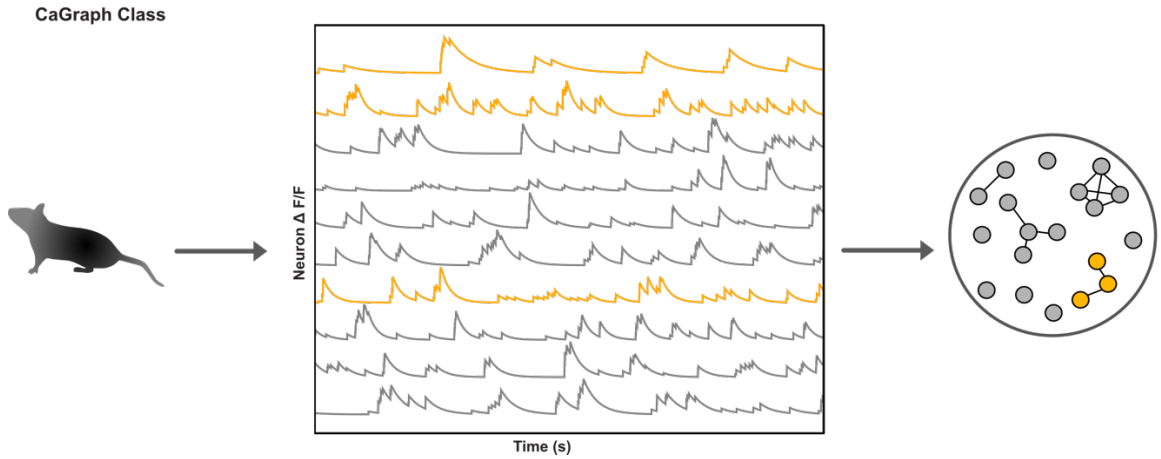

Figure 5: CaGraph class

```
cagraph_obj = CaGraph(data='dataset.csv')
```

2.3.3 *Adding node metadata to a CaGraph object*

In many cases, it may be helpful to incorporate additional data or metadata about the nodes. Previous analyses may have identified groups or clusters of neurons within the dataset that meet specific criteria. For example, one cluster may increase its activity in response to a stimulus while the other decreases its response. Each neuron in this scenario would receive an integer value to identify if it increased (1) or decreased (0) its activity in response to the stimuli. This metadata could be included when the CaGraph object is constructed, allowing the clusters to generate reports that exclusively refer to the subset of identified cells.

```
import numpy as np
METADATA = np.loadtxt('metadata.csv', delimiter=',')
cagraph_obj = CaGraph(data='dataset.csv',
                      node_metadata={'metadata_key': METADATA})
metadata = cagraph_obj.metadata_key
```

2.3.4 *Graph theory analysis examples with the CaGraph object*

There is an extensive list of graph theory analyses that are conceptually possible to compute, many of which are supported in the NetworkX package. However, only a subset of these analyses which are particularly relevant to the study of functional neuronal networks will be supported in cagraph V1.0.0. These include degree distributions, graph density, clustering coefficient distributions, correlated pair ratio distributions, hub identification and betweenness centrality scores, and community identification using greedy modularity maximization.

```
cagraph_obj = CaGraph(data='dataset.csv')

clustering = cagraph_obj.graph_theory.get_clustering_coefficient()
hubs = cagraph_obj.graph_theory.get_hubs()
communities = cagraph_obj.graph_theory.get_communities()
density = cagraph_obj.graph_theory.get_density()
```

2.3.5 *CaGraph reports*

The CaGraph reports that are generated can be returned as a pandas.DataFrame object for further analysis in Python. It can also be saved as a CSV file to be analyzed or visualized in external software.

```
report = cagraph_obj.get_report()
```

	hubs	degree	clustering coefficient	communities	correlated pair ratio	betweenness centrality
0	0	7	0.380952	4	0.179487	0.068709
1	0	4	0.333333	0	0.102564	0.043538
2	0	3	0.666667	0	0.076923	0.002371
3	0	0	0.000000	6	0.000000	0.000000
4	0	7	0.428571	2	0.179487	0.049102
...	...	...	...	...	...	...
34	1	8	0.357143	2	0.205128	0.145899
35	1	5	0.200000	1	0.128205	0.142034
36	0	0	0.000000	12	0.000000	0.000000
37	0	6	0.600000	2	0.153846	0.030712
38	1	5	0.400000	2	0.128205	0.157918

Figure 6: Base report for the CaGraph class.

The base report generated for the CaGraph class contains a set of analyses computed on nodes in the graph. The standard analyses are degree, hubs, clustering coefficient, communities, correlated pair ratio, and betweenness centrality. Each row in the report dataset represents a node in the graph. Continuous metrics, such as the betweenness centrality or clustering coefficient, represent each node with a floating point value. Boolean metrics have a zero when the condition does not apply to the node and a one when the condition does apply. An example is the hub analysis, in which a hub node has a value of one in the report. Metrics that assign nodes to an arbitrary number of groups, such as the communities metric, will be represented with an integer value where each unique integer is a distinct group.

The report can include additional analyses automatically generated when node metadata is included during object construction. For example, suppose the user specifies an attribute called “context_active” at the time of CaGraph construction on a dataset that can take on values of zero,

one, or two to represent the selectivity of the neuron for distinct behavioral contexts. In that case, it will be added to the report.

```
cagraph_obj = CaGraph(data=data,
                node_metadata={'context_active': CONTEXT_METADATA})

report = cagraph_obj.get_report()
```

	hubs	degree	clustering coefficient	communities	correlated pair ratio	betweenness centrality	context_active
0	0	7	0.428571	4	0.179487	0.066371	1.0
2	0	3	0.666667	2	0.076923	0.002466	1.0
4	0	7	0.476190	0	0.179487	0.054889	1.0
6	0	5	0.300000	2	0.128205	0.057955	1.0
7	0	1	0.000000	5	0.025641	0.000000	1.0
...	...	...	...	...	...	...	...
29	0	2	0.000000	3	0.051282	0.018255	1.0
33	0	4	0.166667	1	0.102564	0.043187	1.0
34	0	8	0.357143	0	0.205128	0.128939	1.0
36	0	0	0.000000	11	0.000000	0.000000	1.0
38	1	5	0.400000	0	0.128205	0.146176	1.0

Figure 7: Report processed with user-supplied metadata parsing.

The user can parse the nodes contained in the report by passing the identities of the neurons of interest. This will provide a reduced report.

```
cagraph_obj = CaGraph(data=data,
                node_metadata={'context_active': CONTEXT_METADATA})

report = cg.get_report(parsing_nodes=[1,4,15,38])
```

	hubs	degree	clustering coefficient	communities	correlated pair ratio	betweenness centrality	context_active
1	0	6	0.466667	0	0.153846	0.040791	0.0
4	0	7	0.476190	0	0.179487	0.054889	1.0
15	0	0	0.000000	7	0.000000	0.000000	0.0
38	1	5	0.400000	0	0.128205	0.146176	1.0

Figure 8: Report processed with user-supplied neuron index parsing.

The user can also parse the results using the analyses to create conditions that select only a subset of nodes. For example, the user could request to include neurons with a clustering coefficient greater than 0.3 in the graph. The report will be reduced to only this subset of highly clustered neurons, which can be independently analyzed.

```
cagraph_obj = CaGraph(data=data,
                      node_metadata={'context_active': CONTEXT_METADATA})

report = cagraph_obj.get_report(parse_by_attribute='clustering coefficient',
                                parsing_operation='>',
                                parsing_value=0.2)
```

	hubs	degree	clustering coefficient	communities	correlated pair ratio	betweenness centrality	context_active
0	0	7	0.428571	4	0.179487	0.066371	1.0
1	0	6	0.466667	0	0.153846	0.040791	0.0
2	0	3	0.666667	2	0.076923	0.002466	1.0
4	0	7	0.476190	0	0.179487	0.054889	1.0
6	0	5	0.300000	2	0.128205	0.057955	1.0
...	...	...	...	...	...	...	...
28	0	2	1.000000	1	0.051282	0.000000	0.0
32	0	6	0.466667	0	0.153846	0.047627	0.0
34	0	8	0.357143	0	0.205128	0.128939	1.0
37	0	7	0.619048	0	0.179487	0.033413	0.0
38	1	5	0.400000	0	0.128205	0.146176	1.0

Figure 9: Report processed with built-in graph theory analysis value parsing.

2.3.6 *CaGraphTimeSamples*

The time-sampling graph theory analysis class, CaGraphTimeSamples, provides additional functionality to readily sample the provided dataset using a series of indices that specify the beginning and end of each period of interest. Breaking up the dataset in this way can be useful when the behavioral experiment design incorporates explicit time periods during which the animal was exposed to distinct stimuli or environments. While the time periods could be small in practice, using this class for larger time periods is recommended to reduce spurious correlations.

The user can also choose to automatically generate the series of indices by specifying a rule that must be followed to sample the time series. For example, the user can set the *bin_size* parameter to some value, and the time series will automatically be split into N bins where:

$$N = \frac{length(timeseries)}{bin\_size}$$

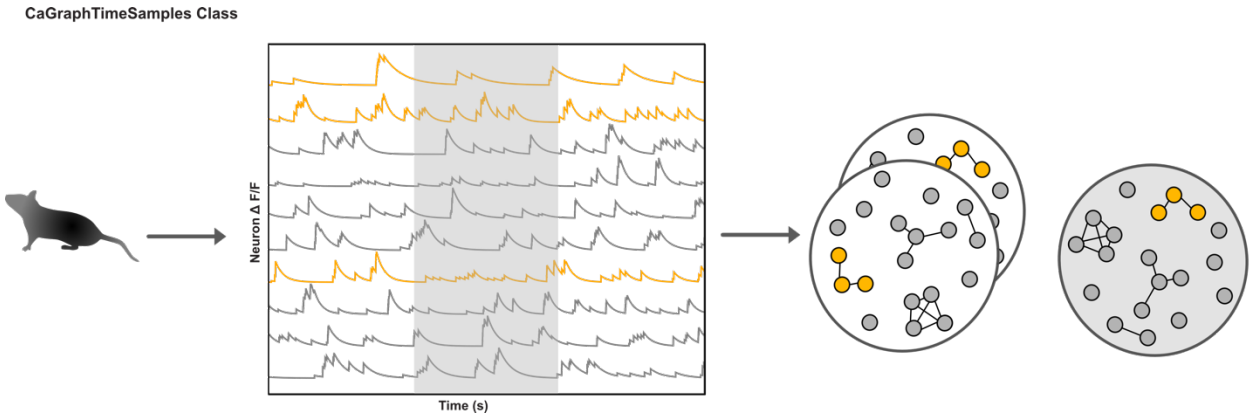

Figure 10: CaGraphTimeSamples class

```
from cagraph import CaGraphTimeSamples

cagraphtimesamples_obj = CaGraphTimeSamples(data='dataset.csv',
                                time_samples=[(0,1800), (1801,3600)],
                                condition_labels=['context_B', 'context_A'])
cagraphtimesamples_obj.get_report()
```

	context_B_hubs	context_A_hubs	context_B_degree	context_A_degree	context_B_clustering coefficient	...	context_A_communities	context_B_correlated pair ratio
0	0	1	7	9	0.476190	...	2	0.057377
1	0	0	3	10	0.666667	...	2	0.024590
2	0	1	3	7	1.000000	...	3	0.024590
3	1	1	9	10	0.138889	...	0	0.073770
4	0	0	4	4	0.500000	...	5	0.032787
...	...	...	...	...	...	...	...	...
117	0	0	8	5	0.321429	...	0	0.065574
118	0	0	5	10	0.200000	...	2	0.040984
119	0	0	7	5	0.476190	...	0	0.057377
120	0	0	6	5	0.533333	...	5	0.049180
121	0	0	3	2	0.666667	...	7	0.024590

Figure 11: Report for the CaGraphTimeSamples class.

The full report generated for the CaGraphTimeSamples class contains all specified graph theory metrics computed for each node and each time sample is iterated over to compute each metric, such that each column contains the metric value for a single time sample condition. However, it is trivial to parse individual CaGraph objects for each time sample by specifying the identifier for the time sample of interest. In this case, the individual reports for each time sample can be provided by selecting a single file or requesting all individual reports.

```
cagraphtimesamples_obj.get_cagraph('context_A').get_report()
```

2.3.7 *CaGraphBatch construction example*

The CaGraphBatch class allows the processing of multiple datasets using a single object instance. When the CaGraphBatch constructor is called, the user must provide a path to a directory containing multiple CSV files of calcium fluorescence recordings. Each file in the directory will be iterated over to create a unique CaGraph object for the dataset, which will be incorporated into the report and can be accessed using an attribute for the dataset identifier.

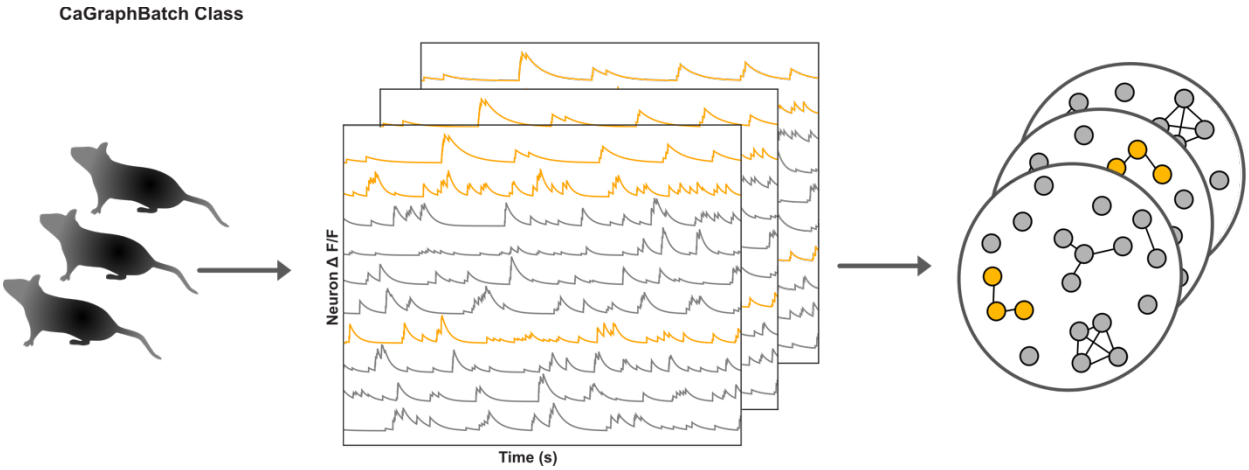

Figure 12: CaGraphBatch class

The full report for the CaGraphBatch class contains the specified graph theory metric for each dataset in the batch. Each column corresponds to a distinct dataset-metric pair, and the identity of the dataset is noted in the column index. As in the CaGraphTimeSamples case, it is trivial to access the CaGraph objects for each dataset using the dataset identifier.

```
from cagraph import CaGraphBatch

cagraphbatch_obj = CaGraphBatch(data_path='../datasets/')
cagraphbatch_obj.get_full_report()
```

	1055-4_D1_hubs	122-2_D1_hubs	14-0_D1_hubs	396-1_D1_hubs	396-3_D1_hubs	...	1055-2_D1_betweenness centrality
0	0.0	1.0	0.0	0.0	0.0	...	0.053713
1	0.0	0.0	0.0	0.0	0.0	...	0.000000
2	0.0	0.0	0.0	0.0	0.0	...	0.000000
3	0.0	0.0	0.0	0.0	0.0	...	0.003381
4	0.0	0.0	0.0	0.0	0.0	...	0.023706
...	...	...	...	...	...	...	...
160	NaN	NaN	NaN	NaN	NaN	...	0.007625
161	NaN	NaN	NaN	NaN	NaN	...	0.015076
162	NaN	NaN	NaN	NaN	NaN	...	0.000000
163	NaN	NaN	NaN	NaN	NaN	...	0.003325
164	NaN	NaN	NaN	NaN	NaN	...	0.030301

Figure 13: Report for the CaGraphBatch class.

Individual reports can be generated by accessing each dataset in the batch by passing the identifier for the dataset of interest.

```
cagraphbatch_obj.get_cagraph('122-2_D1').get_report()
```

2.3.8 *CaGraphBatchTimeSamples construction example*

The CaGraphBatchTimeSamples class combines the functionality of the CaGraphBatch and the CaGraphTimeSamples classes to split multiple datasets within a condition, or batch, into discrete time intervals of interest.

```
from cagraph import CaGraphBatchTimeSamples
cagraphbatchtimesamples_obj = CaGraphBatchTimeSamples(data_path='../datasets/',
                                time_samples=[(0, 1800), (1801, 3600)],
                                condition_label=['context_B', 'context_A'])
```

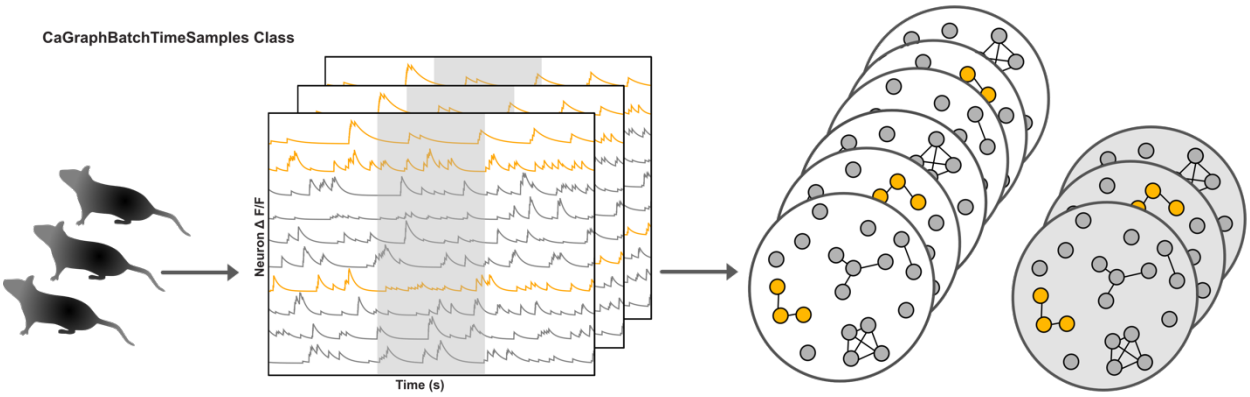

Figure 14: CaGraphBatchTimeSamples class

2.3.9 *CaGraphBehavior construction example*

The CaGraphBehavior class can be used when the experimentalist has a behavioral trace that specifies which behavior an animal was participating in at each time point in the recorded session. To parse the time series appropriately, the user must supply metadata containing integer identifiers for each time point in the dataset. Along with this, the user must provide a map that clarifies the meaning of each integer value. For example, if the behavioral assay tracked whether a mouse was freezing or not freezing in an aversive contextual processing paradigm, each time point that the mouse was frozen would contain a one in the behavioral trace. Each time point that the mouse was not frozen would have a zero in the behavioral trace. Sub-series of the overall behavioral series containing all ones or all zeros would be grouped, and the first and last values in these sub-series would be used to identify the indices at which to split the time series. These indices are used to parse the time series into distinct bins marking the onset and offset of a new integer value and, therefore, a different behavior.

```
from cagraph import CaGraphBehavior
cagraphbehavior_obj = CaGraphBehavior(data_path='dataset.csv',
                                      behavior_data= behavior_data.csv',
                                      behavior_dict={'freezing':1, 'moving':0})
```

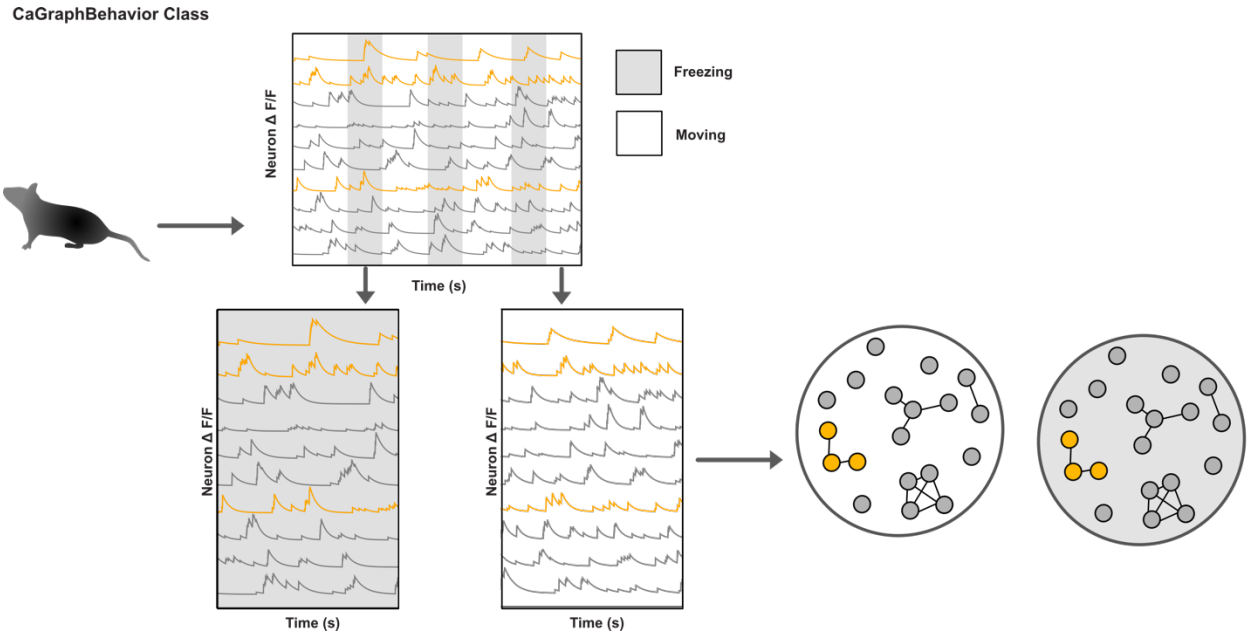

Figure 15: CaGraphBehavior class

2.3.10 *CaGraphMatched construction example*

The CaGraphMatched class can be used in cases where the user has multiple recording sessions of the same field of view and has matched neurons between recording sessions. This class is most useful for behavioral paradigms that require recordings to be made over multiple days, as it is likely that following such longitudinal recordings will manifest with changes to the neurons that are recorded in each session as a result of drift in the field of view, crosstalk due to the activity of neighboring cells, and changes to the cells that are active during the recording session [43]. This drift requires cells to be identified, or registered, across days to compare the same local connectivity over time. By performing this analysis with registered neurons, it is possible to track changes to local topology, which adds a rich set of analyses that are not possible with population-level statistics.

To perform analysis of matched datasets, the user must provide the pair of datasets and a sequence of neuron identifier pairs which serve as a map between neurons that were registered

between the datasets. The matching map can be stored as a CSV file, with each row containing two entries that represent neuron identifiers. If there is no matched neuron for a recorded neuron in the row, the second neuron should be denoted with a zero. The user can also provide a sequence of tuples loaded into memory which contains the index of the matched neuron in the first dataset and the index of the matched neuron in the second dataset as follows: [(0, 4), (2, 10), (4, 1), (34, 5)…]. Here, neuron 4 in the second dataset does not have a matching neuron in the first dataset.

The two datasets could have an unequal but similar number of neurons. Still, the default utility will use the full datasets in each case to generate CaGraph objects. Using the full dataset means that the graphs constructed on each dataset may have different nodes and edges. Alternatively, the user can specify that the graphs should be constructed only using the matched cells between the datasets to explicitly compare topologies with an identical set of nodes.

```
from cagraph import CaGraphMatched

cagraphmatched_obj = CaGraphMatched(
                data_list=['dataset_D1.csv', 'dataset_D9.csv'],
                dataset_labels=['Day_1', 'Day_9'],
                match_map='D1_D9_matching_indices.csv')
```

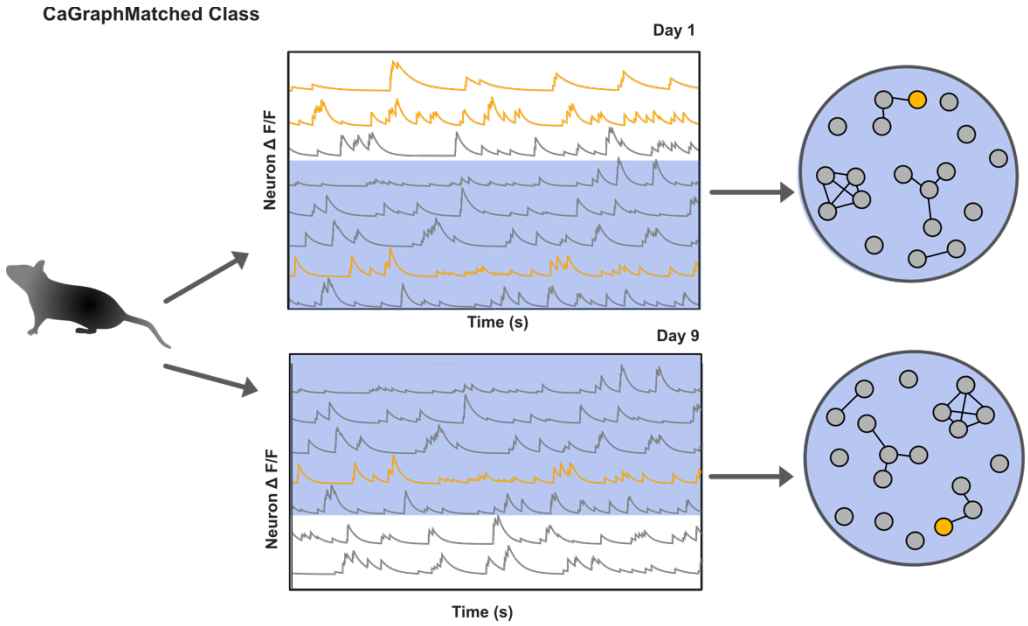

Figure 16: CaGraphMatched class

2.3.11 *Basic interactive visualization with Bokeh*

The cagraph package incorporates the Bokeh package for visualizing interactive HTML graphs, which can be readily inspected using hover tools, zoom, and panning. While NetworkX has some basic visualization tools, it can be complicated to specify and draw graphs that contain sufficient information and are comprehensive, especially for graphs with many nodes. Graphs containing many nodes and edges are typical for analyses of calcium imaging datasets, which often record from hundreds of neurons. Bokeh provides an interactive interface that can scroll, magnify, and hover over nodes and edges to identify information about the graph topology. The cagraph implementation offers a core set of functionalities to hover over and highlight individual nodes and includes attributes about these individual nodes.

Attributes about the nodes include explicit graph theory analyses of individual nodes. The degree, hub identification, clustering coefficient value, correlated pair ratio, and community identification are all incorporated by default for each node. These can be included in the

interactive visualization by specifying hover attributes so that the information about an individual node only appears in the interactive plot when the user hovers over that node.

In addition to specifying hover attributes, the user can choose to adjust the node color or size by the attributes included in the graph. If changing the color, the user can access a wide variety of existing palettes or specify unique palettes manually.

The user can provide a position using an existing NetworkX layout. Still, by default, the graph will use an optimized spring layout that adjusts the location of nodes such that those more densely connected are grouped more closely in the layout. Additionally, it can be helpful to return the position of a graph to use in subsequent visualizations, particularly when analyzing the same network under distinct experimental conditions.

2.3.12 *Incorporating additional node metadata with Bokeh*

In addition to requesting additional graph theory analyses be included in the visualization, it is possible to add node metadata that was attained independently from the graph theory analysis. For example, if cells in the recorded networks were processed in a behavioral paradigm where the animals were exposed to two distinct contexts, context A and context B, cells could be selective for either of the two contexts or not selective for either context, as demonstrated by Seo et al. [44]. This information could be included in the graph visualization by adding node metadata where each node has some integer value representing the identity of the groups (0 – A, 1 – B, 2 – N.S.). Once added as additional attributes, the user could also use this attribute to color the graph, to readily identify whether these predetermined groups also have interesting topology in the network.

```
from cagraph import CaGraph
import visualization as viz
import numpy as np

CONTEXT_METADATA = np.loadtxt('context_metadata.csv', delimiter=',')
cagraph_obj = CaGraph(data='dataset.csv',
                      node_metadata={'context':CONTEXT_METADATA})

palette = ('red','blue','grey') # 0 - A, 1 - B, 2 - N.S.

# Visualize graph with node metadata
viz.interactive_network(cagraph_obj=cagraph_obj,
                        additional_attributes={'context':CONTEXT_METADATA},
                        adjust_node_size_by='degree',
                        adjust_node_color_by='context')
```

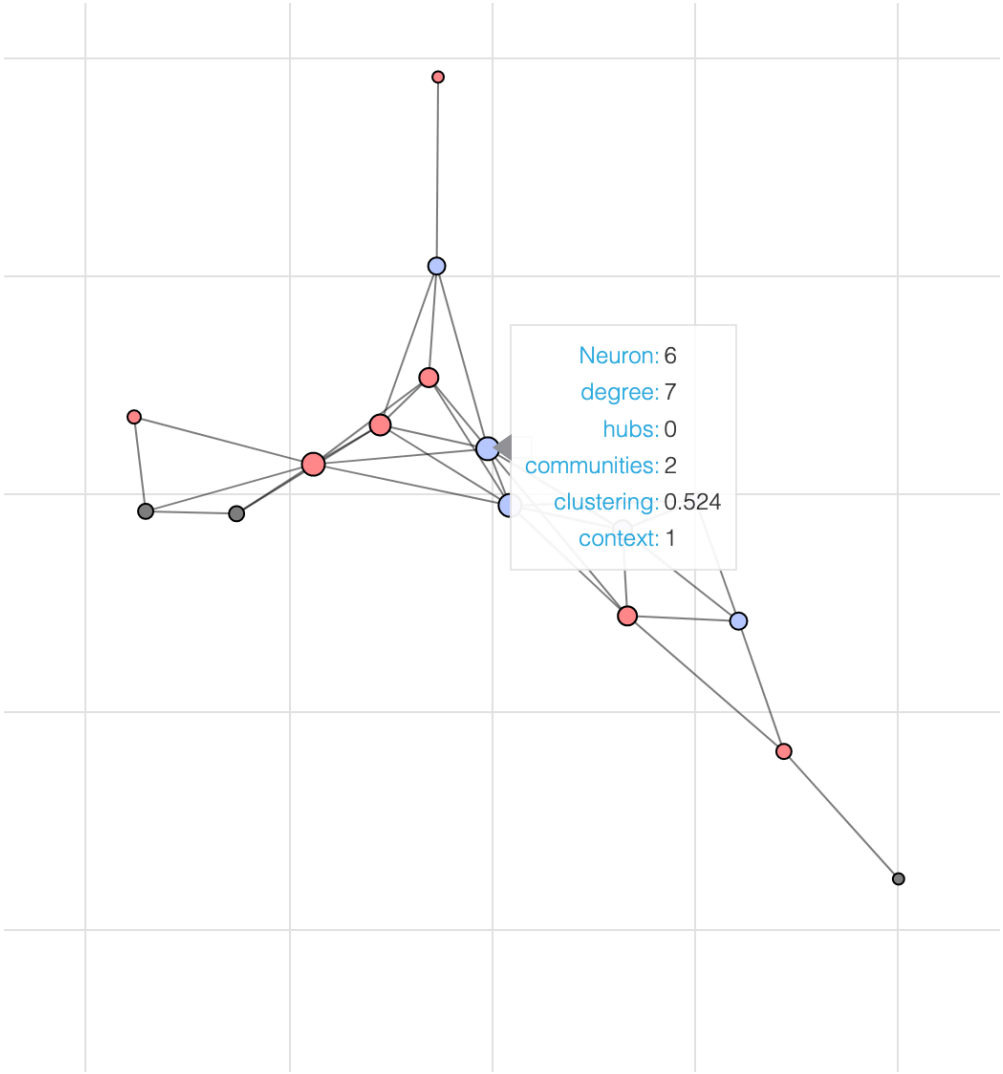

Figure 17: Example interactive visualization with hover attributes.

2.4 Discussion

2.4.1 *Utility of the cagraph package*

cagraph is an open-source Python package for constructing graphs representing functional connectivity of neural dynamics collected with calcium imaging. The package allows for a complex analysis of the topology of local neural microcircuits using graph theory. cagraph provides preprocessing tools to check that the provided calcium imaging dataset is suitable for analysis using undirected and unweighted graphs and visualization tools to inspect the graph and, thereby, the relationships between neurons in the network. The package contains multiple classes for handling batched analyses. These analyses split single datasets into distinct periods sampled from the time series, analyses of matched or tracked datasets in which a subset of cells are consistent across days or contexts, and analyses of behavior-sampled datasets in which the time series is sampled according to the behaviors identified over the recording session. This package allows neuroscientists to analyze functional graphs of calcium imaging data while including critical information about the behavioral experiments under which the neural dynamics were recorded.

2.4.2 *Expanding the biological relevance of the cagraph package*

Weighted and directed graphs

The existing version of the cagraph package does not provide the functionality to include weighted or directed interactions in graphs. The weighted graph option would be beneficial in cases with uncertainty about the recommended threshold for constructing graphs, as it would allow all correlation coefficients to be incorporated directly into the graph. Additionally, it would be possible to include two thresholds for datasets with significant anticorrelations compared to

the shuffled data, adding both an upper and lower threshold and including weights that indicate whether the edges are due to positive or negative correlations. While directed graphs may not be the most common use of graph theory analyses on calcium imaging datasets, which have inherent properties that reduce the ability to extract directed relationships between biological neurons, there are methods to infer these relationships. Transfer entropy is one method that could extract such directed neural connections [45] and could be implemented to construct an interaction matrix in future software updates.

Multi-region graphs

While the examples demonstrated and discussed analyze functional graphs of only a single dataset from a defined local neuroanatomical region of the brain, it would be trivial to extend the software to include distinct subgraphs representing multiple brain regions in a single graph. Multi-region graphs would be useful for cases where recordings from multiple brain regions were taken during a single behavioral session.

Additional metrics for computing correlations

Beyond Pearson's correlation coefficient, additional correlation metrics could be used. These include time-lagged cross-correlations, dynamic time warping [46], and instantaneous phase synchrony [47], to name a few. The software can be extended to include these additional correlation metrics when constructing the functional graphs.

2.5 Conclusion

The cagraph Python package offers neuroscientists a comprehensive analysis environment to assess the quality of their calcium imaging dataset for graph theory analysis, construct functional graphs of neural dynamics and analyze their topology, and visualize these graphs in an interactive framework. The package is readily installable from the PyPi package server, providing a one-line command to install the package and use it in any Python integrated development environment.

2.6 Acknowledgements

The author would like to acknowledge Dr. Sean Piantadosi and Eric Zhang, members of the Bruchas laboratory for providing the datasets used for these graph theory analyses, and for meeting regularly with the author to discuss experimental details which constrained the analyses to improve their biological relevance. The author would like to acknowledge Dr. Charles Zhou and the NAPE Center for providing training in the experimental and data analysis methods of calcium imaging through the 2022 Calcium Imaging Workshop.

2.7 Author contributions

All behavioral and surgical experiments analyzed were conceptualized and performed by Dr. Sean Piantadosi and Eric Zhang, members of the Bruchas laboratory at the University of Washington. All Python code, documentation, and packaging of the cagraph package was conceptualized and developed by Veronica Porubsky. All graph theory analyses of the calcium imaging datasets were performed by Veronica Porubsky.

2.8 CODE AVAILABILITY

The cagraph package is available on PyPi at:

https://pypi.org/project/cagraph/

The formal documentation is under development at:

https://cagraph.readthedocs.io/en/latest/index.html

The GitHub repository is open-source and available at:

https://github.com/vporubsky/CaGraph

Chapter 3. CASE STUDY OF THE ANXIETY CIRCUITRY USING GRAPH THEORY ANALYSES

Two datasets recorded from regions implicated in anxiety are analyzed in relevant behavioral paradigms to demonstrate the utilities available in the cagraph package within a motivating biological context. Anxiety disorders are the most prevalent category of mental health disorders, estimated to affect 28.8% of the United States adult population during their lifetime [48]. Without intervention, many patients develop comorbidities [49]–[51], and pharmacologic treatments for anxiety often lack long-lasting improvement [52] leaving a dire need for new therapeutic designs in treatment-refractory cases. Anxiety disorders and other psychiatric conditions are increasingly viewed as dysregulation of underlying neural circuits [53]–[56]. Computational analyses and modeling offer a rich toolbox to dissect these circuits further to understand the underlying

mechanism governing the encoding of anxiogenic stimuli or to identify topologies that could serve as a biomarker that indicates the healthy or aberrant processing of the anxiogenic stimuli.

Aversive contextual processing paradigms have been used to model behavioral aspects of anxiety disorders, measuring characteristics like lack of exploration in anxiety-provoking or neutral contexts [57]–[60] and dysregulated pattern separation in the dentate gyrus (DG) has been shown to occur in this paradigm, leading to overgeneralization of stimuli characteristic of anxiety [61]. The circuit between the locus coeruleus (LC), dentate gyrus (DG), and the basolateral amygdala (BLA), along with influence from other structures like the medial prefrontal cortex, is implicated in the abnormal aversive contextual processing response observed in anxiety disorder models [61]–[65]. Recent studies propose deep brain stimulation to target aberrant circuits to improve anxiety symptoms, and rodent models support the efficacy of this approach in driving proper aversive contextual processing [66]–[71]. However, it is essential that the mechanisms of encoding anxiogenic stimuli first be elucidated and a biomarker of anxiety in this circuit be identified. Analyzing functional graphs of the neural dynamics in the circuit could reveal patterns of neural activity that are indicative of the underlying encoding mechanism. The LC, DG, and BLA are selected as the regions of interest due to extensive research into their involvement in anxiety disorders [65].

3.1 METHODS

3.1.1 *Surgical procedures*

The surgical procedures used in this work have been previously described [44]. Mice received intracranial injections of specific AAVs to induce the expression of specific proteins required for calcium imaging and chronic intracranial GRIN lens implants to record neural activity by

imaging calcium indicator fluorescence. All implant surgeries were survival surgeries. Groups of 8–12-week mice (with 8-12 mice/group) were prepared. Neural recordings from the dentate gyrus and basolateral amygdala were taken in independent groups. Each group was injected with the viral construct, AAV5-CamKII-GCaMP6f or AAVDJ-Syn-DIO-GCaMP6s, in the region of interest (DG or BLA, respectively), and surgical implantation of the GRIN lens was performed for simultaneous calcium imaging during behavioral studies.

AAV5 was chosen due to its high GFP expression profile in neurons [72]. Stereotaxic surgery and intracranial injection of the viral particles were performed as described previously [73]. Mice were anesthetized in an induction chamber (4% isoflurane) and placed in a stereotaxic frame. They were maintained at 1–2% isoflurane during the procedure. A craniotomy was performed, and mice were injected with 1000 nL of AAV-GCaMP6f unilaterally into the LC (AP: -5.44 mm, ML: +1.25 mm, DV: -3.8 mm), dorsal DG (AP: -2.15 mm, ML: +1.4 mm, DV: -2.1), or BLA (AP: -1.3 mm, ML: +2.9 mm, DV: -4.9 mm). All animals used in these experiments were anesthetized with pentobarbital and transcardially perfused with 4% paraformaldehyde before their brains were harvested, cryoprotected, sectioned, and anatomically verified to confirm injection site. The GRIN lenses for imaging fluorescent activity of the GCaMP6f or GCaMP6s sensors were implanted, and the miniscope was mounted and secured to the GRIN lens following recovery from surgery [74]. Mice recovered for six weeks before beginning aversive contextual processing or OFT or EZM testing to permit optimal AAV expression.

3.1.2 *Calcium imaging methods*

Genetically-encoded calcium indicators (GECIs) can be used in live and behaving neural systems to measure neural activity [75]. One and two-photon calcium imaging can provide cellular-scale resolution images of this neural activity. GCaMP6 [76] is typically used to detect single action

potentials. It is crucial to consider the decay dynamics of the genetically-encoded calcium indicator when analyzing the neural dynamics using both graph theory and recurrent neural network models. Even within the GCaMP family of calcium indicators, there are multiple versions of the optimized sensors, including sensors with slow and fast dynamics. These may impact the underlying correlation structure in the applied analyses discussed. Both fast and slow versions of the GCaMP6 sensor were tested with the cagraph package.

The LC-DG and LC-BLA datasets described in this work were collected using 1-photon calcium imaging, enabling the animals to perform freely moving behaviors. The calcium imaging data was pre-processed and denoised according to standard practices [77], [78], and the denoised trace was used for the subsequent graph theory analyses to reduce the impact of correlations due to noise. The GCaMP6f sensor was excited using 473 nm wavelength light in the LC-DG dataset. The mice received alternating 5 ms pulses of 10 Hz photo-stimulation during trials. The recordings were sampled at 10 Hz. The GCaMP6s sensor was used in the LC-BLA dataset, and the recordings were sampled at 10 Hz.

3.1.3 *Behavioral methods*

Aversive contextual processing paradigm

The calcium imaging recordings used in this thesis were collected in an aversive contextual processing paradigm. This paradigm was studied as a behavioral model of anxiety, in which the freezing of animals was tracked over time. A higher amount of time spent in freezing indicates an increased anxiety response. Aversive contextual processing studies followed a standard protocol for anxiety research [61]. Conditioning took place in two distinct contexts. Neural activity traces were recorded for 197 seconds while the mice were in their home cage prior to training each day. Aversive contextual discrimination training took place in the chamber. In the aversive context

(context A), animals were allowed to freely explore for 180 seconds, after which they received a single 2-second foot shock of 0.50 mA. Mice were taken out of the aversive context chamber 15 seconds after the foot shock and returned to their home cage. In the neutral context (context B), mice explore the context freely for 197 seconds and will then be returned to their home cage. No foot shock was delivered. The aversive contextual discrimination training was repeated for nine days. All mice were placed in the aversive context on the first training day and received a foot shock. In the following days, mice were placed in the neutral context, then moved to the aversive context after 3 hours between neutral context exposure. Sessions were recorded, and videos were scored for freezing to evaluate anxious behavior.

Open field test

The basolateral amygdala recordings used in this thesis were collected in two distinct behavioral tests. The first test – the open field test - allows the experimenter to examine the location of the mouse in an open field during the time course recording. This test can broadly examine animal behavior as they behave freely in the open environment. As such, it can be used to investigate anxiety-like phenotypes by characterizing the time spent in specified behaviors, such as freezing or remaining in particular locations in the open field [79]. Mice in an anxious state will avoid the center of the open field and spend more time at the exterior along the walls. The mice can be tracked using a camera placed above the open field throughout the recording, with the open field parceled into center and exterior regions to quantify the time spent in the two locations.

Elevated zero maze

The basolateral amygdala recordings were also collected in a second test – the elevated zero maze. The elevated zero maze is a behavioral assay similar to the open field test, as it examines the location of the mouse over time. The elevated zero maze consists of an annular platform with four distinct regions, two opposing regions are closed, and two are open [80]. Mice in an anxious state will avoid the open portion of the elevated zero maze and will spend more time in the closed portion. Again, the mice can be tracked using a camera placed above the elevated zero maze to quantify the time spent in the open and closed arms of the maze.

5 Hz stimulation of the LC-BLA noradrenergic projections

The BLA dataset included in this work uses a 5 Hz LC-BLA_{NE} terminal stimulation protocol which has previously been demonstrated to induce an anxiety-like phenotype in mice [81]. This method stimulates norepinephrine projections from the locus coeruleus to the basolateral amygdala.

Statistical power calculations

For the behavioral studies included in this analysis as case studies, power analysis (computed using the G*power software [82]) indicates that a minimum of 8 animals is necessary, as a sample size of 8 in each group will have 99% power to detect this difference in means, assuming a standard deviation of 0.75 using a two-group t-test with a 0.05 two-sided significance level. (This data is typical of place aversion behavior measures such as the proposed freezing behavior to be recorded). Therefore, in the example data used for the analyses of the cagraph package, eight animals were used for behavioral testing with two extra animals to accommodate

for possible losses. This sample size should allow us to achieve adequate power while minimizing the number of animals used in the study to reduce the loss of animal life.

3.1.4 *Assessing statistical significance between distributions*

The Kolmogorov-Smirnov (K-S) test [42] was applied using the SciPy statistics package to assess whether there were statistically distinct distributions between conditions in the clustering coefficient analyses shown. For graph theoretical analyses using cell-matched metadata, which tracks the same cell over successive days of contextual discrimination tasks, paired-samples t-tests [83] were used to determine if there is a significant difference between the means of each connectivity metric, comparing paired recordings from individual subjects in the aversive and neutral contexts. Statistical significance between two conditions is determined by a p-value < 0.05 in the K-S test and paired-samples t-test.

3.2 GRAPH THEORY ANALYSIS OF THE LC-BLA 5 HZ STIMULATION DATASET

The locus coeruleus-norepinephrine system is connected to the anxiety circuitry [84]. The LC is known to alter amygdala and dentate gyrus activity through noradrenergic projections [85], which can be accompanied with an increase in anxious behaviors. The basolateral amygdala plays a role in anxiety circuitry, exhibiting hyperactivity in response to anxiogenic stimuli [86]–[91]. It has been shown previously that stimulation of the LC-BLA noradrenergic projections increases anxiety-like behavior [92].

The elevated zero maze (EZM) [93]–[95] and open field test (OFT) [95], [96] can be used to assess behaviors indicative of anxiety in mice. The BLA dataset described provides a useful case study for the batched, time-sampled graph analysis. The BLA dataset consists of two distinct but similar behavioral paradigms used to study anxiogenic behavior – the elevated zero

maze and the open field test. The study presented in the data shared here used a 5 Hz LC-BLA_{NE} stimulation protocol, in which mice received this stimulation of the norepinephrine terminals projecting from the LC to the BLA, which is known to induce an anxiogenic phenotype in mice. Each dataset in the both the EZM and the OFT batches include ten minutes of recordings made prior to the animal receiving stimulation, ten minutes of recordings made during the stimulation, and ten minutes of recordings made after the animal has received stimulation. During each of these periods, the neural activity and behavior was recorded in the two both behavioral paradigms. The functional connectivity can be studied on these datasets using the CaGraphBatchTimeSamples class.

Figure 17.A shows the stimulation and recording site for the LC-BLA dataset. Figure 17.B shows the behavioral paradigms of the OFT and EZM, in which the LC-BLA datasets are recorded. The stimulation conditions are noted. Behavioral data is shown for each stimulation period in 17.C and 17.D, demonstrating the aversion of animals to the center of the OFT and the open arms of the EZM. The animals prefer to remain in the exterior of the OFT and the closed arms of the EZM. The CaGraphBatchTimeSamples class was then used to generate the subsequent analysis, where three samples were generated for the period before the animal received the stimulation (pre-stim), the period during which the subject received a 5 Hz stimulation of the LC-BLA terminals (stim) and a period following the delivery of stimulation (post-stim) periods.

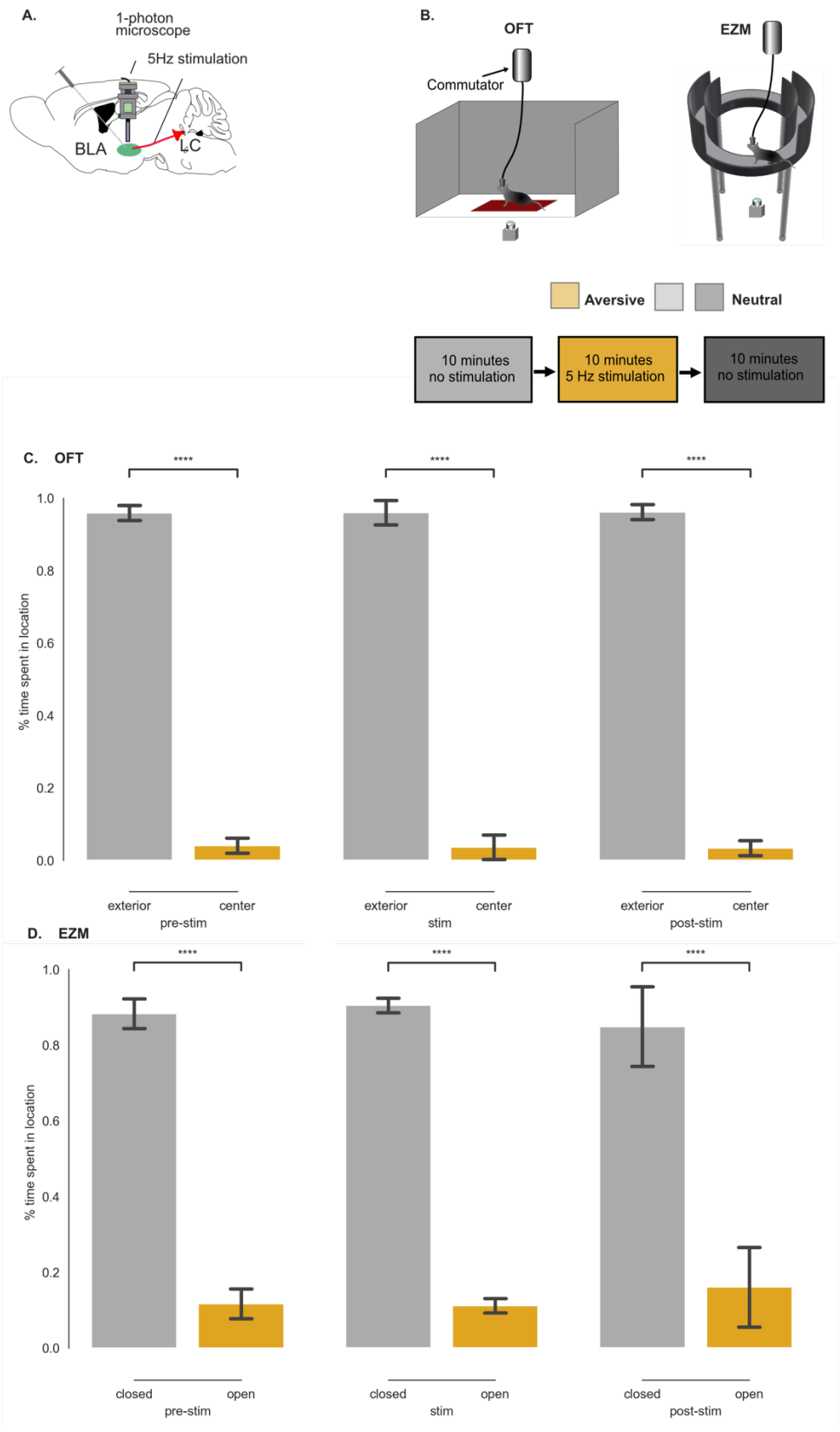

Figure 18: LC-BLA OFT and EZM behavioral data.

Each dataset was sampled according to these specified stimulation periods such that graphs could be constructed only on data recorded during that time period. This allows for the comparison of the distinct graphs using various graph theory metrics. Here, each subject has a corresponding CaGraph object which can be used to study functional connectivity patterns. An initial analysis in Table 1 and Table 2 shows that in both the OFT and the EZM, there is low variability across subjects within each stimulation period, but the mean number of edges and the graph density increases in the 5 Hz stimulation condition. This indicates that neuron pairs are more highly correlated when the anxiogenic stimulation is applied. Further, Figure 19.A and 19.B show the distinct visualizations of the graphs for the pre-stimulation and 5 Hz stimulation conditions, respectively. Note the increase in the number of edges in the 5Hz stimulation graph. Figure 19.C shows pairs of cumulative distribution functions for the clustering coefficient distributions computed on the graphs for each stimulation period, for a single mouse. The 5 Hz stimulation condition is significantly different than the pre-stimulation. Statistical significance is assessed with the KS-test. Further, in Figure 19.D, the average clustering coefficient for each mouse is plotted in the pre-stimulation and 5 Hz stimulation conditions, and a paired samples t-test with a significance value of 0.05 was used to. assess statistical significance between conditions. The 5 Hz stimulation condition shows an increased average clustering across mice.

Table 1: OFT graphs overview.

	Threshold	Nodes	Edges Pre-stim	Edges 5Hz Stim	Edges Post-stim	Graph Density Pre-stim	Graph Density 5Hz Stim	Graph Density Post-stim
519-0	0.115	61.0	384.0	450.0	301.0	0.21	0.246	0.164
120-0	0.267	192.0	1964.0	2223.0	2099.0	0.107	0.121	0.114
568-4	0.159	72.0	327.0	316.0	282.0	0.128	0.124	0.11
658-0	0.13	42.0	184.0	303.0	163.0	0.214	0.352	0.189
568-5	0.159	98.0	671.0	786.0	677.0	0.141	0.165	0.142
120-1	0.185	121.0	1084.0	1164.0	1069.0	0.149	0.16	0.147
119-0	0.248	173.0	1725.0	1882.0	1859.0	0.116	0.126	0.125
651-0	0.187	66.0	397.0	349.0	337.0	0.185	0.163	0.157
mean	0.181	103.0	842.0	934.0	848.0	0.156	0.182	0.144
std	0.053	55.0	680.0	755.0	757.0	0.042	0.08	0.027
median	0.172	85.0	534.0	618.0	507.0	0.145	0.162	0.144
sem	0.019	19.0	240.0	267.0	268.0	0.015	0.028	0.009

Table 2: EZM graphs overview.

	Threshold	Nodes	Edges Pre-stim	Edges 5Hz Stim	Edges Post-stim	Graph Density Pre-stim	Graph Density 5Hz Stim	Graph Density Post-stim
519-0	0.116	61.0	382.0	444.0	298.0	0.209	0.243	0.163
120-0	0.265	192.0	1980.0	2244.0	2121.0	0.108	0.122	0.116
568-4	0.154	72.0	351.0	337.0	298.0	0.137	0.132	0.117
658-0	0.13	42.0	184.0	303.0	163.0	0.214	0.352	0.189
568-5	0.157	98.0	680.0	788.0	687.0	0.143	0.166	0.145
120-1	0.182	121.0	1101.0	1182.0	1087.0	0.152	0.163	0.15
119-0	0.251	173.0	1699.0	1856.0	1831.0	0.114	0.125	0.123
651-0	0.188	66.0	394.0	346.0	334.0	0.184	0.161	0.156
mean	0.18	103.0	846.0	938.0	852.0	0.158	0.183	0.145
std	0.054	55.0	677.0	755.0	757.0	0.041	0.078	0.025
median	0.169	85.0	537.0	616.0	510.0	0.148	0.162	0.148
sem	0.019	19.0	239.0	267.0	268.0	0.014	0.028	0.009

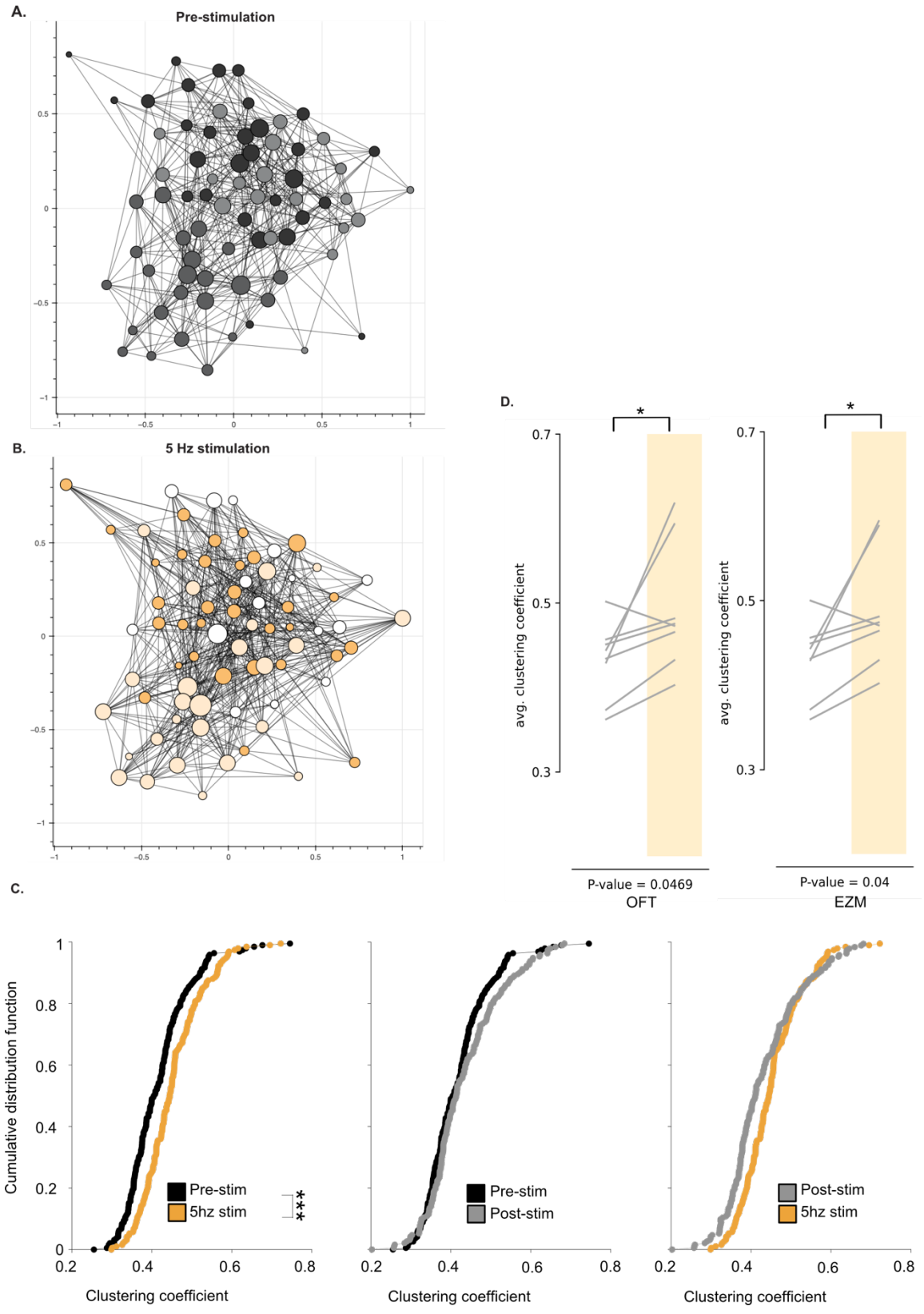

Figure 19: Overview of the LC-BLA dataset graph theory analysis.

3.3 Graph theory analysis of the LC-DG aversive contextual processing dataset

The dentate gyrus also plays a role in anxiety [85], [97]–[99]. The DG, responsible for context encoding and memory acquisition and retrieval [100], [101] has dense expression of adrenergic receptors and inputs from the LC [102] which are known to impair aversive contextual processing of novel neutral contexts in the dorsal DG when stimulated [61]. Additionally, the DG is responsible for pattern separation, and it has been demonstrated that impaired pattern separation occurs in anxiety, resulting in overgeneralization of novel stimuli [103]–[106].

Aversive contextual processing in rodent models has been used to model symptoms of anxiety [57]–[60], manifesting in behavioral characteristics like avoidance [60]. Aversive contextual processing is frequently used to model the underlying fear learning and extinction that occurs in anxiety disorders in both animals and humans and is well-established in mouse models [85], [107], [108]. Behaviors such as time spent freezing can be assessed to determine whether extinction has occurred and can be used to demonstrate the impaired extinction of aversive responses to neutral stimuli. Rodent models of anxiety are well-established and typically seek to recapitulate hypervigilance and avoidance after exposure to an aversive stimulus, which are hallmark behaviors that aid in the clinical diagnosis of anxiety [109], [110].

Table 3: WT Day 1 graphs overview.

	Threshold	Nodes	Edges Context A	Edges Context B	Graph Density Context A	Graph Density Context B
1055-4	0.15	93.0	442.0	468.0	0.103	0.109
122-2	0.123	36.0	49.0	49.0	0.078	0.078
14-0	0.138	39.0	111.0	62.0	0.15	0.084
1055-2	0.33	165.0	633.0	667.0	0.047	0.049
122-3	0.128	43.0	113.0	100.0	0.125	0.111
122-1	0.143	16.0	36.0	26.0	0.3	0.217
1055-3	0.146	122.0	403.0	419.0	0.055	0.057
1055-1	0.276	46.0	61.0	55.0	0.059	0.053
mean	0.179	70.0	231.0	231.0	0.115	0.095
std	0.078	52.0	228.0	249.0	0.083	0.055
median	0.144	44.0	112.0	81.0	0.09	0.081
sem	0.028	18.0	81.0	88.0	0.029	0.019

Table 4: WT Day 9 graphs overview.

	Threshold	Nodes	Edges Context A	Edges Context B	Graph Density Context A	Graph Density Context B
1055-3	0.154	111.0	310.0	303.0	0.051	0.05
1055-1	0.24	49.0	87.0	70.0	0.074	0.06
122-3	0.126	60.0	282.0	218.0	0.159	0.123
122-1	0.182	32.0	127.0	52.0	0.256	0.105
1055-2	0.336	142.0	442.0	490.0	0.044	0.049
1055-4	0.135	107.0	647.0	619.0	0.114	0.109
14-0	0.137	45.0	54.0	145.0	0.055	0.146
122-2	0.114	43.0	86.0	128.0	0.095	0.142
mean	0.178	74.0	254.0	253.0	0.106	0.098
std	0.075	40.0	209.0	205.0	0.072	0.04
median	0.146	54.0	204.0	182.0	0.084	0.107
sem	0.027	14.0	74.0	73.0	0.025	0.014

Table 5: Th-Cre$^{LC\text{-}DG}$ Day 1 graphs overview.

	Threshold	Nodes	Edges Context A	Edges Context B	Graph Density Context A	Graph Density Context B
396-1	0.112	37.0	92.0	107.0	0.138	0.161
396-3	0.18	63.0	103.0	158.0	0.053	0.081
2-1	0.132	43.0	150.0	242.0	0.166	0.268
349-2	0.138	355.0	4347.0	3617.0	0.069	0.058
2-3	0.111	78.0	806.0	576.0	0.268	0.192
387-4	0.169	31.0	78.0	69.0	0.168	0.148
348-1	0.18	32.0	60.0	34.0	0.121	0.069
2-2	0.117	105.0	640.0	1223.0	0.117	0.224
386-2	0.148	45.0	40.0	46.0	0.04	0.046
mean	0.143	88.0	702.0	675.0	0.127	0.139
std	0.028	103.0	1395.0	1168.0	0.07	0.08
median	0.138	45.0	103.0	158.0	0.121	0.148
sem	0.009	34.0	465.0	389.0	0.023	0.027

Table 6: Th-Cre$^{LC\text{-}DG}$ Day 9 graphs overview.

	Threshold	Nodes	Edges Context A	Edges Context B	Graph Density Context A	Graph Density Context B
348-1	0.21	25.0	38.0	35.0	0.127	0.117
2-2	0.118	137.0	1587.0	1813.0	0.17	0.195
386-2	0.108	10.0	3.0	2.0	0.067	0.044
387-4	0.146	31.0	62.0	110.0	0.133	0.237
2-1	0.14	50.0	229.0	173.0	0.187	0.141
349-2	0.135	221.0	1777.0	1358.0	0.073	0.056
2-3	0.108	98.0	997.0	935.0	0.21	0.197
396-1	0.119	38.0	76.0	117.0	0.108	0.166
396-3	0.177	67.0	97.0	118.0	0.044	0.053
mean	0.14	75.0	541.0	518.0	0.124	0.134
std	0.034	67.0	717.0	677.0	0.057	0.071
median	0.135	50.0	97.0	118.0	0.127	0.141
sem	0.011	22.0	239.0	226.0	0.019	0.024

Figure 20.A shows the DG and LC injection sites, and Figure 20.B shows the aversive contextual processing paradigm. The Th-Cre^{LC-DG}::HM3Dq mice included, are those in which the noradrenergic projections from the LC are stimulated to drive anxious behavior. Figure 20.C shows heatmaps of the Pearson's correlation coefficient matrices constructed using a single animal in contexts A and B on day one and day nine. For this mouse, there are distinct patterns in the two contexts on each days. Additionally, it should be noted that there are more neurons recorded on day nine, indicating that small adjustments to the field of view over time, or neurons being active on different days during a longitudinal study, can change the number of neural traces that are deconvolved and extracted for analysis. Figure 20.D shows the cumulative distribution functions for the two contexts on day one and day nine for a single mouse. The observed change in clustering between the two contexts in this mouse is not representative of a pattern across all mice. In the final panel Figure 20.E, the average clustering coefficients for all mice are shown. There are no statistically significant differences found between the average clustering coefficients of the mice in context A and context B in either the WT or Th-Cre^{LC-DG} conditions, assessed using a paired samples t-test with a significance value of 0.05.

Figure 21.A shows a comparison of the mean clustering coefficient for the WT and Th-Cre^{LC-DG} conditions on day one with the mean clustering coefficient of randomized graphs that have been generated from their corresponding ground truth graph using the Maslov and Sneppen algorithm [40]. In each case, the ground truth graph has higher average clustering than the corresponding randomized graph, indicating an inherent structure to the functional connectivity of the neural data. Baseline data recorded before the onset of the aversive contextual processing paradigm was also analyzed. Figure 21.B shows heatmaps of baseline data recorded for a WT and Th-Cre^{LC-DG} mouse, while Figure 21.C shows the threshold plot for these datasets, and

across all mice, the shuffled distribution was not distinguishable from the ground truth data. Further, Figure 21.D shows that the graphs constructed with baseline data have significantly less clustering than their corresponding ground truth datasets recorded on day one and context A. Together, these analyses of the baseline dataset indicate that aversive contextual processing activates the DG and generates meaningful correlation structure which could contribute to the encoding of fear memories.

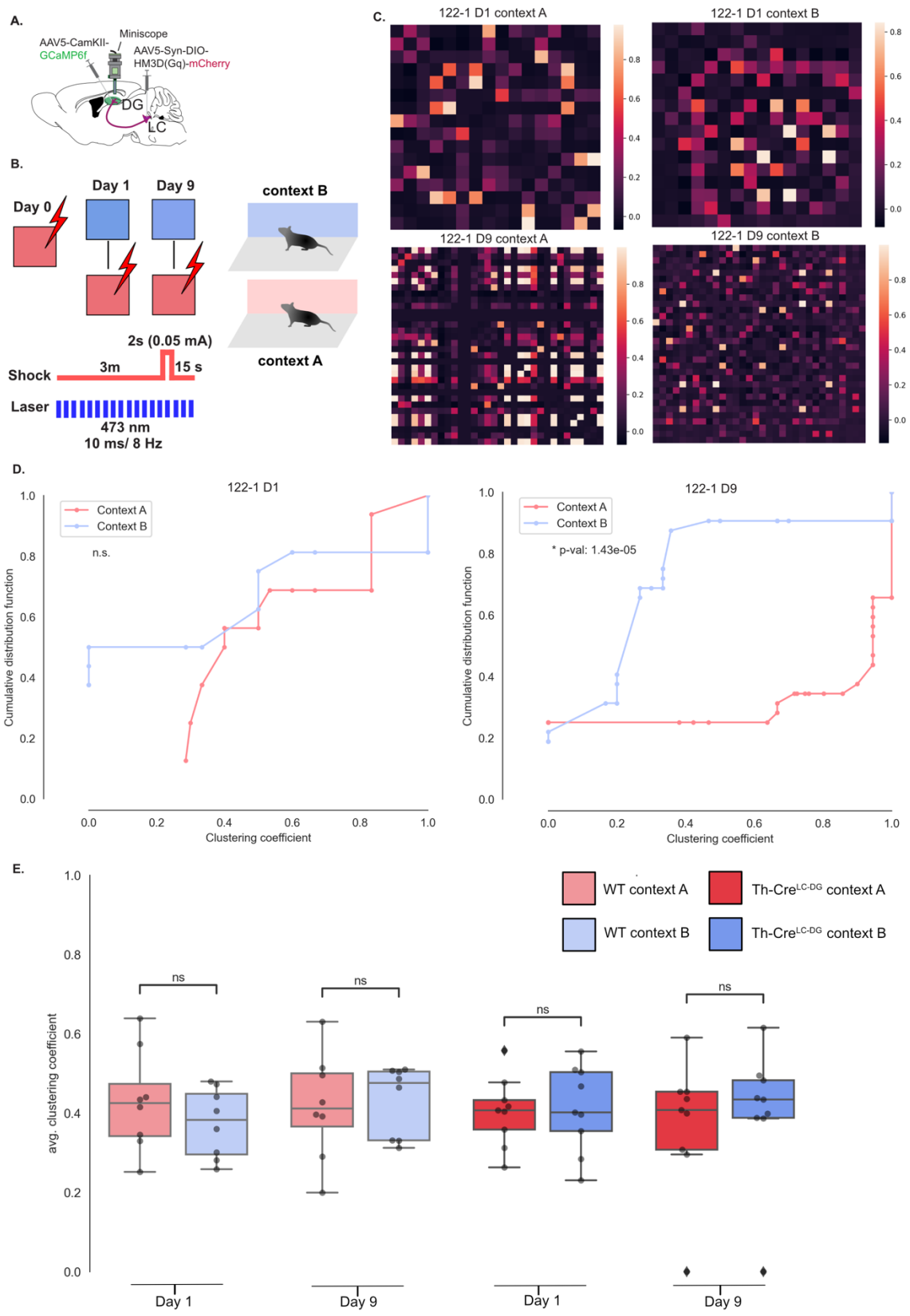

Figure 20: Preliminary analysis of the LC-DG aversive contextual processing dataset.

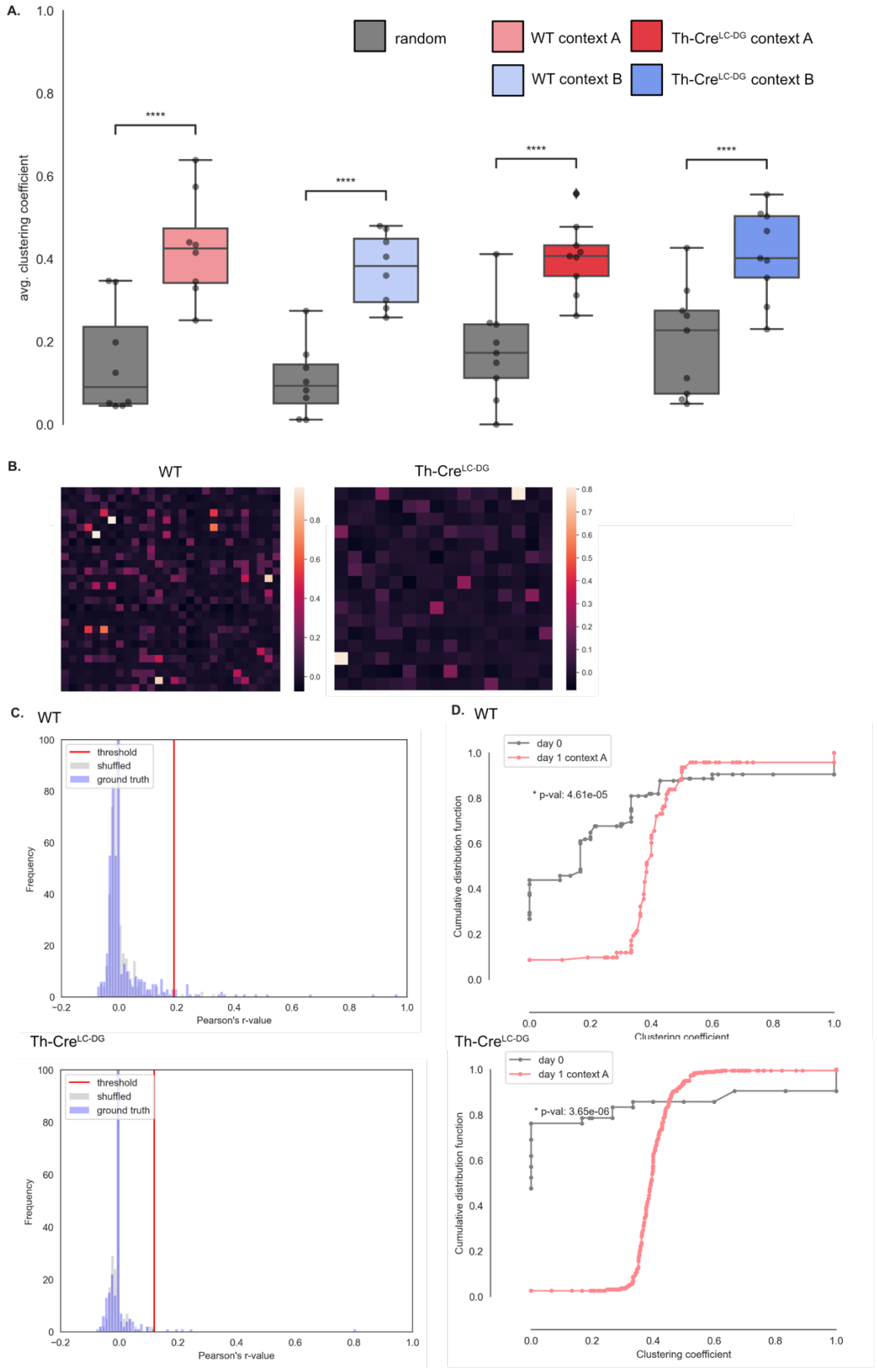

Figure 21: LC-DG comparison to random graphs and baseline data.

Using context selectivity tags, identified in a previous analysis [85], the graphs were parsed to show the clustering coefficient values for only neurons which were selective for context A or context B. Graphs were constructed for context A and context B dynamics for each mouse and the clustering coefficient was computed for each node in the graph. The analysis for mouse 122-1 is shown using the interactive visualization available in the visualization module of the cagraph package in Figure 22.

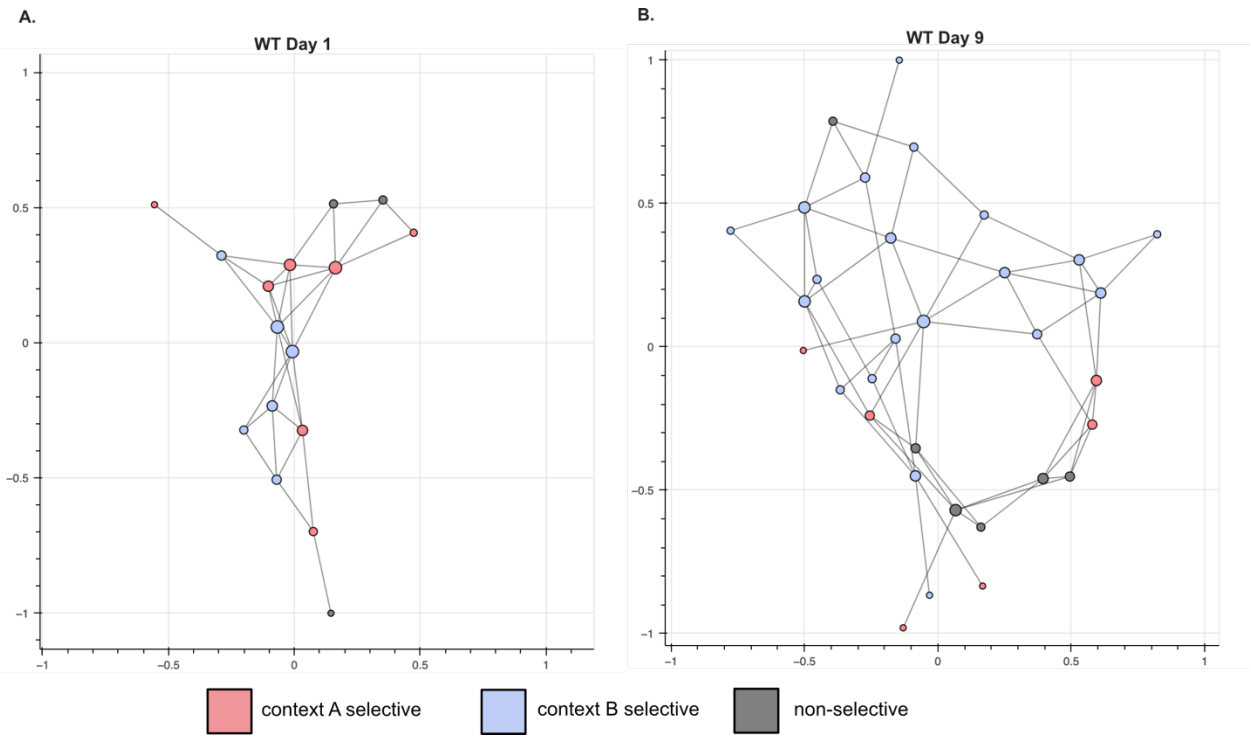

Figure 22: Interactive visualizations of the LC-DG dataset using metadata parsing.

3.4 DISCUSSION

3.4.1 *Implications for studies of the anxiety circuitry*

LC-BLA dataset

The results analyzing the clustering coefficient distributions of the LC-BLA dataset across stimulation periods suggest an increase in synchrony of the recorded neural ensemble while the norepinephrine projections from the LC to the BLA are stimulated at 5 Hz. This result could indicate that the concerted firing of the BLA neurons is driven by norepinephrine neuromodulation by the LC in response to anxiogenic stimuli. Additional studies of the graph topology will be performed to understand how coordinated activity in the BLA is involved in this anxiety response.

LC-DG dataset

Graphs constructed from the observed LC-DG calcium imaging recordings show a distinct correlation structure from randomized graphs and shuffled neural data. In addition to this distinction from random data, the observed LC-DG activity during aversive contextual processing significantly differs from data collected before the onset of the aversive contextual processing paradigm. However, no significant changes were observed in studies of the clustering coefficient distributions comparing the two phenotypes. Similarly, no significant differences were observed across days in the aversive contextual processing paradigm. Still, rich analyses can be performed using the graphs that have already been constructed for these data. Previous analyses of this LC-DG dataset showed that distinct groups of neurons are preferentially active in a single context, and a third distinct group is not preferentially active in either context [44]. It is

possible that investigating the topology of these subgroups could reveal unique patterns in the two phenotypes or throughout aversive contextual processing. Alternatively, additional graph theory algorithms could be applied to look at hubs or community structure to determine whether these topologies differ.

3.5 Conclusion

The preliminary analysis of the LC-DG-BLA circuit in the context of anxiety demonstrates the utility of the cagraph package for analyzing the functional connectivity of neural ensembles using calcium imaging data. The LC-BLA dataset revealed distinct changes to the clustering coefficient distributions in the aversive stimulation condition. While the LC-DG dataset did not show changes to the clustering coefficient distribution between phenotypes or across the behavioral paradigm, the graphs constructed with cagraph allow for additional functional connectivity analyses to be readily performed.

3.6 Acknowledgements

The author would like to acknowledge Dr. Sean Piantadosi and Eric Zhang, members of the Bruchas laboratory for providing the datasets used for these graph theory analyses, and for meeting regularly with the author to discuss experimental details which constrained the analyses to improve their biological relevance. The author would like to acknowledge Dr. Charles Zhou and the NAPE Center for providing training in the experimental and data analysis methods of calcium imaging through the 2022 Calcium Imaging Workshop.

3.7 AUTHOR CONTRIBUTIONS

All behavioral and surgical experiments analyzed were conceptualized and performed by Dr. Sean Piantadosi (LC-BLA EZM and OFT datasets) and Eric Zhang (LC-DG aversive contextual processing dataset), members of the Bruchas laboratory at the University of Washington. All Python code, documentation, and packaging of the cagraph package was conceptualized and developed by Veronica Porubsky. All graph theory analyses of the LC-BLA EZM and OFT datasets and of the LC-DG aversive contextual processing dataset were performed by Veronica Porubsky.

Chapter 4. A PYTHON PACKAGE FOR CONSTRUCTING RECURRENT NEURAL NETWORKS TO REPRESENT NEURAL DYNAMICS USING CALCIUM IMAGING

Neural data is complex, dynamic, and high-dimensional, which can make it difficult to readily propose mechanisms by which neural dynamics in independent brain regions give rise to cognition, behavior, and emotion. Recurrent neural networks (RNNs) provide a model that can be trained to represent sequential data like neural activity and behavioral output to study these processes *in silico.* Here, carnn is introduced – an accessible, flexible, and extensible Python package designed to train recurrent neural network models on calcium imaging datasets, with the option of adding behavioral or cognitive tasks. The carnn package will provide functionality to readily convert a single dataset containing calcium imaging recordings into a recurrent neural network that recapitulates the dynamics of the neural population and the dynamics of individual

neurons within the population. The package will provide functionality to not only construct a computational version of the neural dynamics, but to also offer functionality to specify a behavioral output which can be independently trained from the neural data through distinct readout weight units. Development of this package is ongoing, and this chapter will discuss the planned components of the package.

4.1 BACKGROUND

Constructing models of neuronal populations in the brain using artificial neural networks has greatly interested the computational systems neuroscience community [111]–[114]. Artificial neural networks consist of units called neurons which are connected to one another with synaptic weights. These weights can be updated iteratively throughout training to optimize the fit of the artificial neural network to a provided objective. This objective could be neural data, in which case the individual units of the network may represent corresponding neurons from a live and behaving animal brain. The objective could also be a behavioral or cognitive task, where the artificial neural network would be trained to perform the task. Still, the individual units of this network may not represent the actual neural dynamics of a brain region. While there are a variety of architectures available for artificial neural networks, RNNs are particularly good models for representing neural activity and cognitive tasks, as these are both dynamic and change over time. RNNs incorporate recurrent connectivity patterns, which enable them to process sequences by retaining a memory component, information about previous timesteps.

In contrast to mechanistic models, which are governed by a subset of mathematically defined rules, artificial neural networks have the advantage of using architectures with many optimized synaptic weights that allow the model to represent complex systems and harness local

computations within this architecture to learn relevant neural manifolds in a high-dimensional parameter space [112]. This makes them a good method for representing neural systems *in silico* that behave similarly to the recorded data. They can provide a testing platform to guide experimentation *in vivo.*

4.1.1 *Using recurrent neural networks to model neural dynamics*

RNNs are a powerful class of artificial neural networks that can model sequential data. They are a natural choice for modeling neural dynamics collected using one or two-photon calcium imaging techniques. While feedforward neural networks can be trained to easily classify input data and perform simple tasks, training RNNs on sequential data ensures that they capture temporal dependencies. The architecture of these networks incorporates both the feedforward architecture and feedback to previous layers, adding this powerful memory component. Given the inherent dynamic nature of neural activity, capturing these temporal dependencies in a model system is essential. In addition to the feedback or recurrent architecture of RNNs, the training process enables the network to learn representations of the underlying system, which can predict future activity or provide a forecast for the system. Model training is typically performed in a supervised manner, in which the input time sample is mapped to the future state and internal model parameters, such as the connection weights, are updated according to an appropriate loss function, typically involving a comparison of the predicted value to the actual next value in the sequence.

While recurrent neural networks innately can track sequential data, multiple features can be added or adjusted to improve memory of previous events. The long short-term memory (LSTM) [115] can model long-term dependencies in sequential data. LSTMs have been applied broadly to speech recognition, natural language processing, and time series prediction tasks. In

each of these applications, the input data is a sequence, and the output depends on the context of the entire sequence, including those recently passed and those processed much earlier in the sequence. LSTMs are designed to selectively forget or retain information at each time step, allowing them to handle variable-length inputs. With this feature, LSTMs can capture complex patterns in the data. The gated recurrent unit (GRU) [116] is a recurrent neural network model with functionality similar to the LSTM. It uses a gating mechanism to update only a portion of the internal state of the model. Due to this selective update, GRUs have fewer parameters than LSTMs, making them more computationally efficient to train and simulate.

Still, additional methods exist which can be used to reduce the dimensionality of the high-dimensional recorded neural system while constructing a model that maps neural activity to behavior, mood, or state. For example, an approach pioneered by Pandarinath et al. in 2018 called Latent Factor Analysis via Dynamical Systems uses a nonlinear dynamical system to infer latent dynamics of a neurological system underlying the single-trail spiking activity to predict behavioral variables or, conversely, to predict neural dynamics using changes to the recorded behavioral output [117]. This package uses discrete spiking activity to infer the latent dynamical system. However, a similar package that performs LFADS (caLFADS) on calcium imaging data has been designed [118]. This package is not supported with a robust Python library.

Many recurrent neural networks have been used to model neurological systems for experimentation *in silico.* However, a RNN modeling package which can be readily used to reconstruct calcium imaging data without substantial user input required to optimize the model would provide a useful tool for experimentalists to guide experimentation and discovery.

4.1.2 *carnn: a Python package for constructing recurrent neural network models of one- and two-photon calcium imaging data*

Here, the plan for an open-source Python package called carnn is described, designed to analyze calcium imaging data collected with one- and two-photon calcium imaging using recurrent neural network models is described with case study examples demonstrating the utilities of the package. Due to the complexity of existing Python packages designed to construct neural network models, which offer extensive functionality, it can be prohibitive for non-specialists to devote significant time and resources to learning an entirely new set of analyses that require experience in programming, machine learning, and software development. carnn will need minimal details from the user to specify, build, and train a recurrent neural network model to replicate the dynamics of a provided calcium imaging dataset. The package will be accessible and will not require extensive deep-learning knowledge to use. As a result, the user will be able to rapidly construct an RNN and use the model to guide experimentation. The backend of the software will be based on PyTorch [119], a well-supported package for building and training neural network models. This robust infrastructure will ensure that the models created using the carnn package are extensible, allowing for additional customization by researchers familiar with the PyTorch framework. The carnn package will enable the user to train both neural dynamics and behavioral tasks simultaneously, with a set of neurons representing the neuronal units identified in the calcium imaging dataset and an output unit representing the behavioral activity.

RNNs have been used widely to model neural systems, and packages to support this modeling approach already exist. PsychRNN is designed to model behavioral tasks for neuroscience and is built using TensorFlow as the underlying neural network construction infrastructure [120]. As such, it functions similarly to the output unit trained in this carnn

package. This package is easily accessed through the PyPi server and is well-supported. However, it does not incorporate the explicit training of the RNN on neural data and focuses on representing the behavioral data instead. An article by Perich et al. presents an approach called Current-Based Decomposition (CURBD), which employs data-constrained recurrent neural network models to reproduce neural dynamics and infer directional currents between multiple brain regions [121]. This implementation provides functionality beyond replicating only the individual neural dynamics by providing a measure of the interactions between multiple brain regions. However, it does not incorporate the training of behavioral outputs. The CURBD implementation also uses a manual implementation of an RNN, which has been programmed using numerical analysis libraries instead of a more robust library designed to support neural network construction, such as TensorFlow or PyTorch [119]. As mentioned previously, LFADS can be used to construct models of neurological systems [117] and caLFADS is specifically designed to support modeling calcium imaging data [118]. Implementations exist in Python, but may only be suitable for specialists.

It would be useful to construct a package that not only provides neuroscientists with a method to build models of their dynamic neural and behavioral data but is also easy to use and contains specified experiments that can be executed on the constructed models. These approaches will allow well-studied algorithms to be applied to understanding dynamic connectivity patterns [122], [123]. Computational modeling has often struggled to reach translational utility, but blending these computational analyses with wet-lab experiments would present an attempt to surmount these barriers.

4.2 METHODS

The carnn package will be built using several open-source python packages which support mathematical and scientific computing in Python. The basic numerical computations will be performed using including Numpy [24] and scientific computing, including statistical analyses, will be performed using SciPy [25]. The construction of the recurrent neural network models will be handled with PyTorch [119]. For plotting and result visualization, Matplotlib [28], [29] and Seaborn [29] will be used. The reports generated using the software are handled using the Pandas package [30].

4.2.1 *Preprocessing pipeline for one- and two-photon calcium imaging data*

The preprocessing steps required for extracting calcium fluorescence data are described in Chapter 3.

4.2.2 *Input data types to the CaRNN class*

Currently, the CaRNN class supports input data as loaded numpy.ndarray types, CSV files, and simple NWB files, which consist of a single HDF5 file. If the dataset is not loaded into memory as a numpy.ndarray, the full path to the CSV or NWB file must be specified. The input data must be oriented such that the first row of the data represents the time points at which the calcium fluorescence imaging session was sampled. The subsequent rows must include the calcium fluorescence traces for each neuron in the network, sampled at each time point specified in the time row. Then, each column represents the state of the recorded neural population at a single time point.

4.2.3 *Constructing the CaRNN object*

Calcium imaging data will be used to construct CaRNN objects for each dataset. A schematic is shown in Figure 23.A which illustrates that the goal of the CaRNN class is to generate an RNN model that can reproduce the neural dynamics of the original dataset. Figure 23.B shows a schematic for computing the residuals between the observed neural data and predicted model output. Figure 23.C shows how the model is optimized over successive training iterations, updating weights in the network to enable a better fit of the model to the experimental data. Figure 23.D demonstrates an optional output unit containing behavioral data encoded as a vector of integer values representing the possible behavioral states. Figure 23.E shows that training results in an update to the weight distributions in the network, such that the initial random weight distribution no longer aligns with that of the trained network weights.

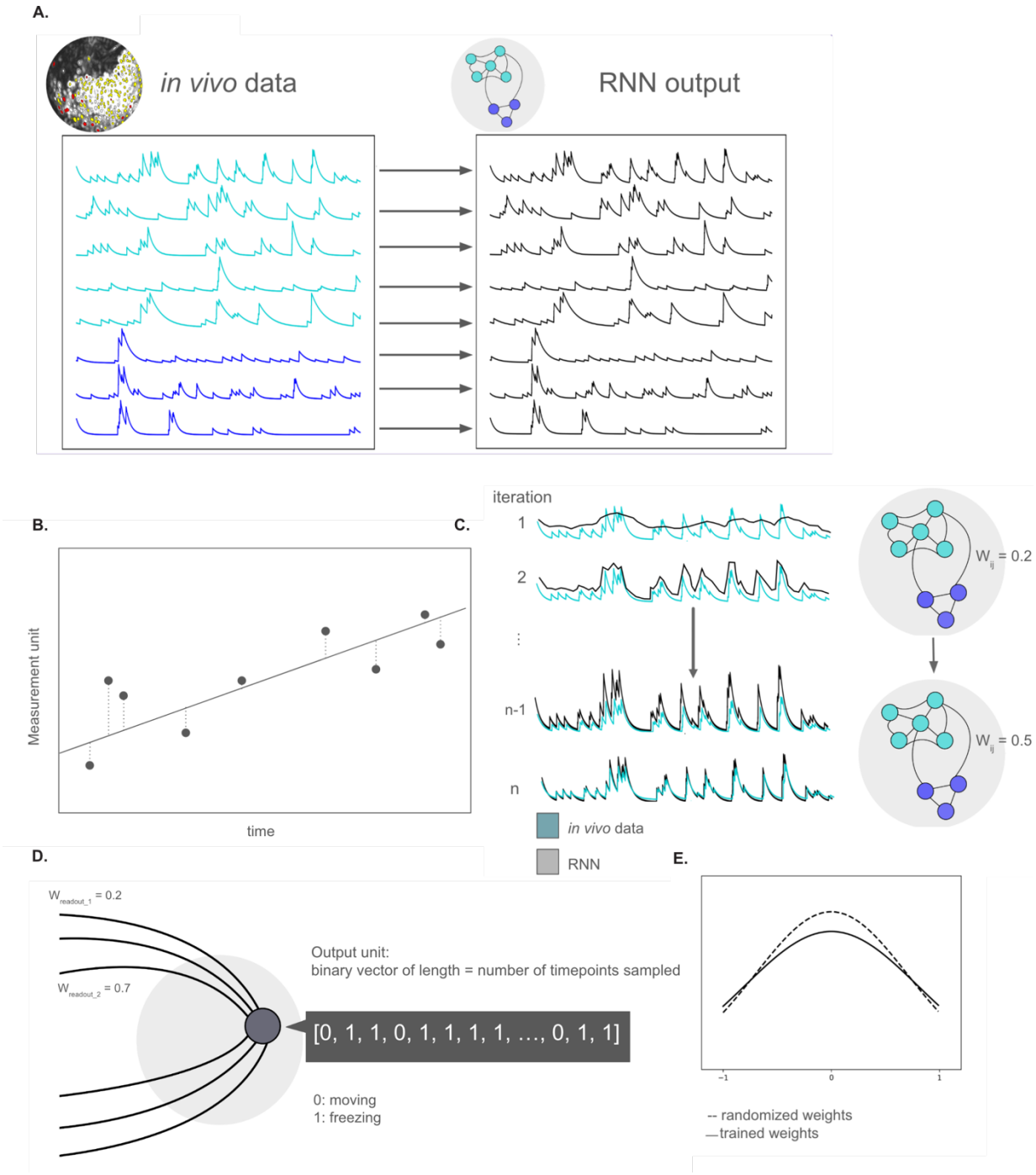

Figure 23: CaRNN schematic.

Initializing network weights and bias values

The weights and bias values will be initialized randomly, and with training, it is expected that the distribution of weights will change.

Objective function

In the planned implementation, the loss, or objective function is the Mean Squared Error (MSE) loss. This will be used to reproduce neural dynamics collected using calcium imaging data by comparing the model output to the observed data in the neural time series. The MSE loss is typically used for regression tasks, as it measures the average squared difference between the predicted output and true values. It penalizes larger deviations more than smaller ones by squaring the difference between the model and the observed data. Additional objective functions will be assessed for biological relevance, for example – if the region of interest is known to exhibit sparse connections between units, the L1 norm could be employed.

Optimization algorithm

The planned optimization algorithm to train the model is Adaptive Moment Estimation (Adam) [124]. It takes the parameters of the network – the weights – and the learning rate as inputs. The Adam optimizer updates the parameters of the network during training to minimize the loss function using gradient descent.

Learning rule

The learning rule that will be used in the first version of the carnn software is a standard backpropagation algorithm, which is an essential component of the optimization process in deep-learning. Backpropagation allows the neural network to update its weights based on the calculated gradients of the loss function with respect to the network parameters [125]. The backpropagation algorithm propagates the gradients backward through the network, starting from

the loss and going through each layer. It applies the chain rule of calculus to compute the gradients layer by layer, efficiently distributing the contribution of each parameter to the overall loss [125]. This enables the network to adjust its weights and biases in a way that minimizes the loss function and improves performance over time.

Default network architecture of a CaRNN object

Constructing a CaRNN object will build a network which has the same number of input units and output units as the number of distinct calcium fluorescence traces in the observed imaging neural time series. The hidden layer sizes to be used in the default architecture will be optimized when testing the implementation of the CaRNN class on sample datasets.

4.2.4 *Training the CaRNN*

During the training loop the loss will computed using the MSE loss function between the observed neural data and the model predictions, and the gradients will be computed using backpropagation. Then, the optimizer will perform the parameter update to all trainable weights in the network.

4.2.5 *Use of LSTM or GRU to enable improved memory of historic data*

While the initial attempts to generate functional RNNs that can reproduce neural data will use standard continuous recurrent neural network architecture, gated recurrent units (GRUs) [115], [126] and long short-term memory units (LSTMs) [115] will also be tested to ensure proper handling of vanishing gradients, and better memory of the historic training data. The package will provide a default RNN architecture to use for non-specialists, but enable more experienced modelers to substitute RNN architectures with a subset of supported networks, or with novel

user-provided architectures with appropriate input and output relationships that fit the specification. Initial attempts to construct RNNs with the carnn package that adequately represent the dynamics of calcium imaging test datasets and predict data that was not included during training have performed poorly, even when GRUs and LSTMs are used in the architecture. A more complex architecture, such as the variational auto-encoder used in the LFADS approach, may be required as the carnn package is developed.

4.2.6 *Running arbitrary time-length predictions from initial conditions or from subset of data*

The RNNs constructed using the CaRNN package should be capable of running arbitrary time-length predictions using either initial conditions or a subset of data. These are not guaranteed to be good representations of the system when predicting for periods of time that extended far beyond than the original training dataset, but could still be executed if desired. Again, incorporating a variational auto-encoder approach as in the LFADS paradigm may allow a reduced representation of the underlying dynamical system to be learned, such that longer predictions are possible.

4.2.7 *Multi-objective optimization with behavioral trace*

After the core network trained on the neural dynamics contained within calcium imaging datasets has been properly optimized, a second objective to train the network to perform classification using a behavior trace could be added. The behavioral trace should be the same length as the neural time series. Each entry in the behavioral trace will be an integer value, where each integer maps to a unique behavior that is observed at each time step. This will require an output unit with distinct readout weights to be included in the network during training. This output unit will have a distinct objective capable of performing a classification task at each time step.

4.2.8 *Evaluating CaRNN performance*

Validation metrics to evaluate performance of the RNNs constructed using the CaRNN class will be added. The loss computed during training will be reported. Experiments will be performed to test how accurately the model predicts output using initial conditions after training.

4.2.9 *Loss function to assess behavioral data fit*

An additional loss function will be considered for studies in which the behavioral data is included in modeling. For example, the RNN will predict the behavioral state of the animal given only the neural data. To accomplish this, the model should also minimize the prediction error of an output classifier unit which outputs a binary list of values encoding the behavioral state at each time point. This has been minimally tested in the carnn package with poor results for the dual-objective training, and requires substantial optimization.

4.2.10 *Assessing population-wide dynamics with principal components analysis*

Global modes of activity from each neuroanatomical region, determined using principal components analysis will be used to check that RNN models can reproduce these emergent properties within a margin of error to be considered a sufficient representation of the system when the model is simulated from initial conditions. Such RNN models which reproduce global or collective network behaviors have been demonstrated to adequately represent the electrophysiological network effects [127]. Simple principal components analysis will be applied to the population activity of the RNN and original neural data. Before performing principal components analysis on neural dynamics, the denoised calcium imaging datasets are z-scored to standardize all time courses to zero mean and unit variance.

4.2.11 *CaRNN base report*

A standardized report method will be available in the carnn package so that any singular dataset which is used to create a CaRNN object can be used to generate a complete overview of the model performance. This report will be exported to a CSV file or analyzed immediately in Python as a Pandas DataFrame object.

4.2.12 *Packaging and testing*

Given that a focus of this package is ease of use and robustness, it is necessary that there are adequate checks for proper use. Currently, input validation is not implemented. This will be updated so that all inputs are thoroughly checked and adequate feedback is provided to the user with error and warning messages when necessary, before proceeding with the analysis. Currently, handling of errors and warnings is largely left up to the dependency packages and may not provide sufficient information to correct the error. To further assist with ease of use, the documentation of the package will be updated to include usage instructions with examples using the package.

Currently, the carnn package has the infrastructure implemented for continuous integration using GitHub Actions, such that a test suite is executed when changes are committed via git. However, the test suite for the package is under development. Tests will be added which use an example dataset to confirm that all functions can be executed as expected and that they return the desired results.

4.3 DISCUSSION

4.3.1 *Utility of the proposed carnn package*

The proposed work provides a design for the carnn package, allowing users to readily construct RNN models of neural dynamics collected with calcium imaging data. With the option to incorporate a multi-objective optimization, behavioral data could also be recapitulated in the proposed modeling paradigm. These models would allow experimental neuroscientists to test their hypotheses *in silico* before proceeding with experiments *in vivo.* With a well-validated model that consistently predicts new neural data and behavior, it may be possible to use these computational experiments to suggest mechanisms of information processing and to understand how complex neural dynamics give rise to cognition, behavior, and emotion.

4.3.2 *Constructing a recurrent neural network for case studies of the anxiety circuitry*

In future work, RNN models will be constructed to examine LC-DG-BLA circuit dysregulation in the context of animal models of anxiety. Populations of model neurons with continuous-time dynamics will be initialized using the LC, DG, and BLA calcium imaging data. Biologically relevant calcium transient rates will be learned through training the RNN. The models for each region in the circuit will then be constructed in Python using the PyTorch package and simulator. Each cell in the calcium imaging data will have a corresponding RNN unit, so each RNN unit takes input training data from calcium imaging experiments and must produce an output that approximates the observed neural dynamics by learning the underlying dynamical neural system. After training, the RNN should reproduce the approximate dynamics of the original experiment given only an initial state for the system – the calcium fluorescence value at the first collected timestamp. Initial tests using the datasets described in Chapter 3 have been performed during the

development of the carnn package, and a more robust architecture will likely be required to provide a suitable model of the neural dynamics recorded with a continuous calcium fluorescence signal. Variational auto-encoders have successfully reconstructed continuous neural signals, as in work by Nolan et al. in 2022, in which this architecture was used to reconstruct electrocorticography signals [128], and it may be useful to pursue a similar architecture to reconstruct calcium fluorescence signals.

4.3.3 *Incorporating biological relevance in RNN models*

Biologically-plausible learning rules and connectivity constraints

While the initial implementation of the carnn software only uses standard backpropagation, which is mathematically robust but not biologically relevant, other biologically plausible learning rules could be added as options in the software. Constraints on the connectivity of the neural network architecture could also be optionally imposed when appropriate. For example, the software could be designed to allow users to specify the probability of connectivity between neurons or to make the weights negative or positive depending on the percentage of inhibitory and excitatory synapses expected in the region of interest. These constraints will further improve the biological plausibility of the model.

Multiple connected regions

Extending this project could result in the addition of multiple connected regions for neural recordings simultaneously recorded from multiple brain regions. The neural activity of each region could be simultaneously optimized as described for the single region case proposed. Creating a multi-region RNN would be a useful feature to allow detailed studies of relevant

neural circuits, integrating activity from multiple regions to give rise to a given behavioral, cognitive, or emotional state.

4.3.4 *Towards translational utility*

A core motivation for the construction of this software was to provide an easy-to-use recurrent neural network constructor for neuroscientists to model neurological microcircuits *in silico*. However, this becomes more useful when the model is combined with additional experiments to validate the model results. Future work with two-photon calcium imaging experiments could include an interface to a spatial light modulator photo-stimulation pattern constructor, which automatically identifies neurons in the network that are highly influential over the behavioral outcome. The experimentalist then could use the RNN to readily identify cells of interest based on predefined computational experiments and use those identified cells to perform some stimulation-induced activity reduction *in vivo*.

4.4 CONCLUSION

The carnn package is currently in the early stages of development and will require substantial development before it is ready for distribution and use. While the carnn package requires the development of the RNN network construction and training processes, the structure of the package at the level of utility and structure will largely resemble the cagraph package, which is more fully developed. Namely, the input datatype checks, the ease of use of the CaRNN class, and the package testing, documentation, and deployment will be carried over from the CaGraph class. The remaining optimization will be primarily confined to the model architecture and multi-objective training paradigm, which will ensure that models can reconstruct the observed calcium imaging and behavioral datasets. If successful, these models can serve as a useful tool to guide

experimentation and investigate the mechanisms by which neural activity gives rise to cognition, behavior, and emotion.

4.1 ACKNOWLEDGEMENTS

The author would like to acknowledge Dr. Joseph Hellerstein and Dr. Lucian Smith, who provided feedback on the proposed RNN modeling approach, analyses of state space, and *in silico* experiments through lab meeting comments and one-on-one discussions. The author would like to acknowledge Dr. Charles Zhou and the NAPE Computing Subgroup, who offered feedback on the feasibility and utility of my proposed work.

4.2 AUTHOR CONTRIBUTIONS

All behavioral and surgical experiments analyzed were performed by Dr. Sean Piantadosi and Eric Zhang, members of the Bruchas laboratory at the University of Washington. All Python code, documentation, and packaging of the carnn package was conceptualized and developed by Veronica Porubsky. All preliminary analyses of calcium imaging datasets using the carnn package were performed by Veronica Porubsky.

Chapter 5. SUMMARY

The primary goal of the research described and proposed in this document is to demonstrate the utility of two Python packages for studying the functional connectivity of neuronal microcircuits recorded using calcium imaging. The studies employed graph theory analyses and modeling with recurrent neural networks to guide experimentation and investigate network topology and underlying mechanistic details. To date, most of the work completed towards these goals is

described in Chapters 2 and 3, which addresses the cagraph Python package developed to analyze the functional connectivity and topology of networks of neurons recorded using calcium imaging and the application of this package to datasets collected from the dentate gyrus and basolateral amygdala. The package is intended to be easy to install and use and offers classes accommodating standard behavioral experiment designs. Most of the functionality for assessing the graphs is performed using the underlying NetworkX package, a powerful library for computing graph theory analyses. By making the underlying graph objects in the cagraph package easy to access, scientists familiar with the NetworkX package can readily extend their analyses by using the graphs constructed in cagraph with the functionality of interest in NetworkX.

The remaining work described in Chapter 4 begins to address the proposed work using recurrent neural networks to create representations of neural dynamics collected using calcium imaging *in silico*. The Python package carnn is intended to be easy to use. The models will be trained to reproduce the dynamics of the calcium imaging input data at single-neuron and population-level resolution. The proposed objective functions for recurrent neural network training will fit the models to individual neural fluorescence traces. A reduced dimensional state space will be used to evaluate the population-level correspondence of the model to the neural data. Such models could guide experimentation by discovering mechanistic details underlying processes. Many of the technical details for this work – such as the exact loss functions to be employed, which functional connectivity constraints will be further optimized, and the network architecture that best replicates the dynamics in the calcium imaging datasets - are still being investigated.

BIBLIOGRAPHY

[1] O. Sporns, "Contributions and challenges for network models in cognitive neuroscience," *Nature Neuroscience*, vol. 17, no. 5. Nature Publishing Group, pp. 652–660, 2014. doi: 10.1038/nn.3690.

[2] B. Bollobás, *Modern graph theory*. Springer. 1998.

[3] C. J. Nelson and S. Bonner, "Neuronal graphs: a graph theory primer for microscopic, functional networks of neurons recorded by Calcium imaging."

[4] E. Bullmore and O. Sporns, "Complex brain networks: Graph theoretical analysis of structural and functional systems," *Nature Reviews Neuroscience*, vol. 10, no. 3. pp. 186–198, Nov. 2009. doi: 10.1038/nrn2575.

[5] J. Z. Kim, J. M. Soffer, A. E. Kahn, J. M. Vettel, F. Pasqualetti, and D. S. Bassett, "Role of graph architecture in controlling dynamical networks with applications to neural systems," *Nat Phys*, vol. 14, no. 1, pp. 91–98, 2018, doi: 10.1038/NPHYS4268.

[6] M. D. Humphries, "Dynamical networks: Finding, measuring, and tracking neural population activity using network science.," *Netw Neurosci*, vol. 1, no. 4, pp. 324–338, 2018, doi: 10.1162/NETN_a_00020.

[7] F. Fröhlich, *Network Neuroscience*. Elsevier Inc., 2016. doi: 10.1038/nn.4502.

[8] J. D. Medaglia, M. E. Lynall, and D. S. Bassett, "Cognitive network neuroscience," *J Cogn Neurosci*, vol. 27, no. 8, pp. 1471–1491, Nov. 2015, doi: 10.1162/jocn_a_00810.

[9] C. J. Stam, "Modern network science of neurological disorders," *Nature Publishing Group*, vol. 15, 2014, doi: 10.1038/nrn3801.

[10] U. Braun, A. Schaefer, R. F. Betzel, H. Tost, A. Meyer-Lindenberg, and D. S. Bassett, "From Maps to Multi-dimensional Network Mechanisms of Mental Disorders," *Neuron*, vol. 97, no. 1. Cell Press, pp. 14–31, Nov. 2018. doi: 10.1016/j.neuron.2017.11.007.

[11] A. Fornito, A. Zalesky, E. Bullmore, *Fundamentals of Brain Network Analysis*, Elsevier Inc*,* 2016, doi: 10.1016/C2012-0-06036-X.

[12] F. V. Farahani, W. Karwowski, and N. R. Lighthall, "Application of graph theory for identifying connectivity patterns in human brain networks: A systematic review," *Front Neurosci*, vol. 13, no. JUN, p. 585, 2019, doi: 10.3389/FNINS.2019.00585/BIBTEX.

[13] O. Sporns, "Cerebral cartography and connectomics," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 370, no. 1668, May 2015, doi: 10.1098/RSTB.2014.0173.

[14] U. Braun, A. Schaefer, R. F. Betzel, H. Tost, A. Meyer-Lindenberg, and D. S. Bassett, "From Maps to Multi-dimensional Network Mechanisms of Mental Disorders," *Neuron*, vol. 97, no. 1, pp. 14–31, Jan. 2018, doi: 10.1016/J.NEURON.2017.11.007.

[15] M. Rubinov and E. Bullmore, "Fledgling pathoconnectomics of psychiatric disorders," *Trends Cogn Sci*, vol. 17, no. 12, pp. 641–647, Dec. 2013, doi: 10.1016/J.TICS.2013.10.007.

[16] D. S. Bassett and E. T. Bullmore, "Human brain networks in health and disease," *Curr. Opin. Neurol.*, vol. 22, pp. 340–347, 2009.

[17] E. Bullmore and O. Sporns, "Complex brain networks: graph theoretical analysis of structural and functional systems," *Nat. Rev. Neurosci.*, vol. 10, no. 3, pp. 186–198, Mar. 2009, doi: 10.1038/nrn2575.

[18] A. Fornito and E. T. Bullmore, "Connectomics: a new paradigm for understanding brain disease," *Eur. Neuropsychopharmacol.*, vol. 25, pp. 733–748, 2015.

[19] E. Pnevmatikakis. “Analysis pipelines for calcium imaging data,” *Current Opinion in Neurobiology*, vol. 55, pp. 15-21, April 2019, doi: https://doi.org/10.1016/j.conb.2018.11.004.
[20] P. Zhou *et al.*, “Efficient and accurate extraction of in vivo calcium signals from microendoscopic video data,” *Elife*, vol. 7, Feb. 2018, doi: 10.7554/ELIFE.28728.
[21] D. O’Connor *et al.*, “Functional network properties derived from wide-field calcium imaging differ with wakefulness and across cell type,” *Neuroimage*, vol. 264,p p. 119735, Dec. 2022, doi: 10.1016/J.NEUROIMAGE.2022.119735.
[22] D. H. Lim, J. M. LeDue, and T. H. Murphy, “Network analysis of mesoscale optical recordings to assess regional, functional connectivity,” *https://doi.org/10.1117/1.NPh.2.4.041405*, vol. 2, no. 4, pp. 041405, Jun. 2015, doi: 10.1117/1.NPH.2.4.041405.
[23] M. Rubinov, O. Sporns, “Complex network measures of brain connectivity: uses and interpretations,” *Neuroimage*, vol. 52, pp. 1059-1069, Sept. 2010, doi: 10.1016/J.NEUROIMAGE.2009.10.003
[24] C. Harris *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, pp. 357-362, Sept. 2020, doi: 10.1038/S41586-020-2649-2
[25] P. Virtanen *et al.*, “SciPy 1.0: fundamental algorithms for scientific computing in Python,” *Nature Methods,* vol. 17, p. 262-272, March 01, 2023, doi: 10.1038/S41592-019-0686-2
[26] A. Hagberg, P. Swart, and D. Chult, “Exploring network structure, dynamics, and function using NetworkX,” *Proceedings of the 7th Python in Science Conference (SciPy2008),* pp. 11-15, Aug. 2008.
[27] Bokeh Development Team, “Bokeh: Python library for interactive visualization.” 2018.
[28] J. H.-C. in science & engineering and undefined 2007, “Matplotlib: A 2D graphics environment,” *Computing in Science & Engineering,* vol. 9, pp. 90-95
[29] M. L. Waskom, “seaborn: statistical data visualization,” *J Open Source Softw*, vol. 6, no. 60, p. 3021, Apr. 2021, doi: 10.21105/JOSS.03021.
[30] W. McKinney, “Data Structures for Statistical Computing in Python,” *Proceedings of the 9th Python in Science Conference*, pp. 56–61, 2010, doi: 10.25080/MAJORA-92BF1922-00A.
[31] E. A. Pnevmatikakis *et al.*, “Simultaneous Denoising, Deconvolution, and Demixing of Calcium Imaging Data,” *Neuron*, vol. 89, no. 2, pp. 285–299, Jan. 2016, doi: 10.1016/J.NEURON.2015.11.037.
[32] J. Friedrich, P. Zhou, and L. Paninski, “Fast online deconvolution of calcium imaging data,” *PLoS Comput Biol*, vol. 13, no. 3, p. e1005423, Mar. 2017, doi: 10.1371/JOURNAL.PCBI.1005423.
[33] E. Kolaczyk and G. Csárdi, “Statistical analysis of network data with R,” *Springer Cham,* 2020, doi: 10.1007/978-3-030-44129-6
[34] D. Freedman, R. Pisani, and R. Purves, *Statistics (international student edition)*, 4th ed. New York: WW Norton & Company, 2007.
[35] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’ networks,” *Nature 1998 393:6684*, vol. 393, no. 6684, pp. 440–442, Jun. 1998, doi: 10.1038/30918.
[36] U. Brandes, “A faster algorithm for betweenness centrality*,” *https://doi.org/10.1080/0022250X.2001.9990249*, vol. 25, no. 2, pp. 163–177, 2010, doi: 10.1080/0022250X.2001.9990249.

[37] H. Beyer, “Tukey, John W.: Exploratory Data Analysis. Addison-Wesley Publishing Company Reading, Mass. — Menlo Park, Cal., London, Amsterdam, Don Mills, Ontario, Sydney 1977, XVI, 688 S.,” *Biometrical Journal*, vol. 23, no. 4, pp. 413–414, Jan. 1981, doi: 10.1002/BIMJ.4710230408.

[38] A. Clauset, M. E. J. Newman, and C. Moore, “Finding community structure in very large networks,” *Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics*, vol. 70, no. 6, p. 6, Dec. 2004, doi: 10.1103/PHYSREVE.70.066111/FIGURES/3/MEDIUM.

[39] A. Clauset, M. E. J. Newman, and C. Moore, “Finding community structure in very large networks,” *Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics*, vol. 70, no. 6, p. 6, Aug. 2004, doi: 10.1103/PhysRevE.70.066111.

[40] S. Maslov and K. Sneppen, “Specificity and stability in topology of protein networks,” *Science*, vol. 296, no. 5569, pp. 910–913, May 2002, doi: 10.1126/SCIENCE.1065103.

[41] Z. Abu-Aisheh, R. Raveaux, J. Y. Ramel, and P. Martineau, “An Exact Graph Edit Distance Algorithm for Solving Pattern Recognition Problems,” *ICPRAM 2015 - 4th International Conference on Pattern Recognition Applications and Methods, Proceedings*, vol. 1, pp. 271–278, Jan. 2015, doi: 10.5220/0005209202710278.

[42] F. J. Massey, “The Kolmogorov-Smirnov Test for Goodness of Fit,” *J Am Stat Assoc*, vol. 46, no. 253, p. 68, Mar. 1951, doi: 10.2307/2280095.

[43] L. Sheintuch *et al.*, “Tracking the Same Neurons across Multiple Days in Ca2+ Imaging Data,” *Cell Rep*, vol. 21, no. 4, p. 1102, Oct. 2017, doi: 10.1016/J.CELREP.2017.10.013.

[44] D. oh Seo *et al.*, “A locus coeruleus to dentate gyrus noradrenergic circuit modulates aversive contextual processing,” *Neuron*, vol. 109, no. 13, pp. 2116-2130.e6, Jul. 2021, doi: 10.1016/j.neuron.2021.05.006.

[45] F. Wörgötter *et al.*, “Transfer Entropy as a Measure of Brain Connectivity: A Critical Analysis With the Help of Neural Mass Models,” *Frontiers in Computational Neuroscience | www.frontiersin.org*, vol. 1, p. 45, 2020, doi: 10.3389/fncom.2020.00045.

[46] D. Berndt and J. Clifford, “Using Dynamic Time Warping to Find Patterns in Time Series,” 1994.

[47] J.-P. Lachaux, E. Rodriguez, J. Martinerie, and F. J. Varela, “Measuring Phase Synchrony in Brain Signals,” *Hum Brain Mapping*, vol. 8, pp. 194–208, 1999, doi: 10.1002/(SICI)1097-0193(1999)8:4.

[48] R. C. Kessler, P. Berglund, O. Demler, R. Jin, K. R. Merikangas, and E. E. Walters, “Lifetime Prevalence and Age-of-Onset Distributions of DSM-IV Disorders in the National Comorbidity Survey Replication,” *Arch Gen Psychiatry*, vol. 62, no. 6, p. 593, Jun. 2005, doi: 10.1001/archpsyc.62.6.593.

[49] A. N. Goldstein-Piekarski, L. M. Williams, and K. Humphreys, “A trans-diagnostic review of anxiety disorder comorbidity and the impact of multiple exclusion criteria on studying clinical outcomes in anxiety disorders,” *Transl Psychiatry*, vol. 6, no. 6, p. e847, 2016, doi: 10.1038/tp.2016.108.

[50] Y. Zhou *et al.*, “Comorbid generalized anxiety disorder and its association with quality of life in patients with major depressive disorder,” *Sci Rep*, vol. 7, no. 1, pp. 1–8, Jan. 2017, doi: 10.1038/srep40511.

[51] F. Lamers *et al.*, “Comorbidity patterns of anxiety and depressive disorders in a large cohort study: The Netherlands Study of Depression and Anxiety (NESDA),” *Journal of Clinical Psychiatry*, vol. 72, no. 3, pp. 342–348, Mar. 2011, doi: 10.4088/JCP.10m06176blu.

[52] B. Bandelow, S. Michaelis, and D. Wedekind, “Treatment of anxiety disorders,” *Dialogues Clin Neurosci*, vol. 19, no. 2, pp. 93–106, 2017, doi: 10.4324/9780203728215-33.

[53] T. Insel *et al.*, “Research Domain Criteria (RDoC): Toward a New Classification Framework for Research on Mental Disorders,” *American Journal of Psychiatry*, vol. 167, no. 7, pp. 748–751, Jul. 2010, doi: 10.1176/appi.ajp.2010.09091379.

[54] J. Sheynin and I. Liberzon, “Circuit dysregulation and circuit-based treatments in posttraumatic stress disorder.,” *Neurosci Lett*, vol. 649, pp. 133–138, 2017, doi: 10.1016/j.neulet.2016.11.014.

[55] S. L. Rauch, L. M. Shin, and E. A. Phelps, “Neurocircuitry Models of Posttraumatic Stress Disorder and Extinction: Human Neuroimaging Research-Past, Present, and Future Functional Neuroimaging Findings in PTSD,” 2006, doi: 10.1016/j.biopsych.2006.06.004.

[56] I. Liberzon and J. L. Abelson, “Context Processing and the Neurobiology of Post-Traumatic Stress Disorder.,” *Neuron*, vol. 92, no. 1, pp. 14–30, Oct. 2016, doi: 10.1016/j.neuron.2016.09.039.

[57] N. Singewald and A. Holmes, “Rodent models of impaired fear extinction.,” *Psychopharmacology (Berl)*, vol. 236, no. 1, pp. 21–32, Jan. 2019, doi: 10.1007/s00213-018-5054-x.

[58] M. R. Milad and G. J. Quirk, “Fear Extinction as a Model for Translational Neuroscience: Ten Years of Progress,” *Annu. Rev. Psychol*, vol. 63, pp. 129–151, 2012, doi: 10.1146/annurev.psych.121208.131631.

[59] S. Lissek *et al.*, “Classical fear conditioning in the anxiety disorders: A meta-analysis,” *Behaviour Research and Therapy*, vol. 43, no. 11, pp. 1391–1424, Nov. 2005, doi: 10.1016/j.brat.2004.10.007.

[60] C. Herry, F. Ferraguti, N. Singewald, J. J. Letzkus, I. Ehrlich, and A. Lüthi, “Neuronal circuits of fear extinction,” *European Journal of Neuroscience*, vol. 31, no. 4. John Wiley & Sons, Ltd, pp. 599–612, Feb. 01, 2010. doi: 10.1111/j.1460-9568.2010.07101.x.

[61] D. Seo *et al.*, “A locus coeruleus to dentate gyrus noradrenergic circuit modulates aversive contextual processing,” *Neuron*, vol. 109, no. 13, pp. 2116-2130.e6, Jul. 2021, doi: 10.1016/J.NEURON.2021.05.006.

[62] J. G. McCall *et al.*, “CRH Engagement of the Locus Coeruleus Noradrenergic System Mediates Stress-Induced Anxiety,” *Neuron*, vol. 87, no. 3, pp. 605–620, Aug. 2015, doi: 10.1016/j.neuron.2015.07.002.

[63] J. G. McCall *et al.*, “Locus coeruleus to basolateral amygdala noradrenergic projections promote anxiety-like behavior,” *Elife*, vol. 6, Jul. 2017, doi: 10.7554/eLife.18247.

[64] A. Wagatsuma, T. Okuyama, C. Sun, L. M. Smith, K. Abe, and S. Tonegawa, “Locus coeruleus input to hippocampal CA3 drives single-trial learning of a novel context,” *Proc Natl Acad Sci U S A*, vol. 115, no. 2, pp. E310–E316, 2017, doi: 10.1073/pnas.1714082115.

[65] P. Tovote, J. P. Fadok, and A. Lüthi, “Neuronal circuits for fear and anxiety,” *Nat Rev Neurosci*, vol. 16, no. 6, pp. 317–331, Jun. 2015, doi: 10.1038/nrn3945.

[66] R. Reznikov, M. Binko, J. N. Nobrega, and C. Hamani, “Deep Brain Stimulation in Animal Models of Fear, Anxiety, and Posttraumatic Stress Disorder,” *Neuropsychopharmacology*, vol. 41, no. 12, pp. 2810–2817, Nov. 2016, doi: 10.1038/npp.2016.34.

[67] B. A. Dengler, S. A. Hawksworth, L. Berardo, I. McDougall, and A. M. Papanastassiou, "Bilateral amygdala stimulation reduces avoidance behavior in a predator scent posttraumatic stress disorder model," *Neurosurg Focus*, vol. 45, no. 2, p. E16, 2018, doi: 10.3171/2018.5.FOCUS18166.

[68] R. Reznikov *et al.*, "Prefrontal Cortex Deep Brain Stimulation Improves Fear and Anxiety-Like Behavior and Reduces Basolateral Amygdala Activity in a Preclinical Model of Posttraumatic Stress Disorder," *Neuropsychopharmacology*, vol. 43, no. 5, pp. 1099–1106, Apr. 2018, doi: 10.1038/npp.2017.207.

[69] R. W. Bina and J.-P. Langevin, "Closed Loop Deep Brain Stimulation for PTSD, Addiction, and Disorders of Affective Facial Interpretation: Review and Discussion of Potential Biomarkers and Stimulation Paradigms.," *Front Neurosci*, vol. 12, p. 300, 2018, doi: 10.3389/fnins.2018.00300.

[70] A. S. Widge *et al.*, "Treating refractory mental illness with closed-loop brain stimulation: Progress towards a patient-specific transdiagnostic approach," *Exp Neurol*, vol. 287, pp. 461–472, 2017, doi: 10.1016/j.expneurol.2016.07.021.

[71] M. M. Shanechi, "Brain–machine interfaces from motor to mood," *Nat Neurosci*, vol. 22, no. 10, pp. 1554–1564, Oct. 2019, doi: 10.1038/s41593-019-0488-y.

[72] H. A. Petrosyan, V. Alessi, V. Singh, A. S. Hunanyan, J. M. Levine, and V. L. Arvanian, "Transduction efficiency of neurons and glial cells by AAV-1, -5, -9, -rh10 and -hu11 serotypes in rat spinal cord following contusion injury," *Gene Therapy 2014 21:12*, vol. 21, no. 12, pp. 991–1000, Aug. 2014, doi: 10.1038/gt.2014.74.

[73] J. G. McCall *et al.*, "Locus coeruleus to basolateral amygdala noradrenergic projections promote anxiety-like behavior.," *Elife*, vol. 6, 2017, doi: 10.7554/eLife.18247.

[74] L. Zhang *et al.*, "Miniscope GRIN Lens System for Calcium Imaging of Neuronal Activity from Deep Brain Structures in Behaving Animals," *Curr Protoc Neurosci*, vol. 86, no. 1, p. e56, Jan. 2019, doi: 10.1002/cpns.56.

[75] T. W. Chen *et al.*, "Ultra-sensitive fluorescent proteins for imaging neuronal activity," *Nature*, vol. 499, no. 7458, p. 295, Jul. 2013, doi: 10.1038/NATURE12354.

[76] T. W. Chen *et al.*, "Ultra-sensitive fluorescent proteins for imaging neuronal activity," *Nature*, vol. 499, no. 7458, p. 295, Jul. 2013, doi: 10.1038/NATURE12354.

[77] E. A. Pnevmatikakis *et al.*, "Simultaneous Denoising, Deconvolution, and Demixing of Calcium Imaging Data," *Neuron*, vol. 89, no. 2, p. 285, 2016, doi: 10.1016/J.NEURON.2015.11.037.

[78] P. Zhou *et al.*, "Efficient and accurate extraction of in vivo calcium signals from microendoscopic video data," *Elife*, vol. 7, Feb. 2018, doi: 10.7554/ELIFE.28728.

[79] M. L. Seibenhener and M. C. Wooten, "Use of the Open Field Maze to measure locomotor and anxiety-like behavior in mice," *J Vis Exp*, no. 96, Feb. 2015, doi: 10.3791/52434.

[80] J. K. Shepherd, S. S. Grewal, A. Fletcher, D. J. Bill, and C. T. Dourish, "Behavioural and pharmacological characterisation of the elevated 'zero-maze' as an animal model of anxiety," *Psychopharmacology (Berl)*, vol. 116, no. 1, pp. 56–64, Sep. 1994, doi: 10.1007/BF02244871.

[81] J. G. McCall *et al.*, "Locus coeruleus to basolateral amygdala noradrenergic projections promote anxiety-like behavior," *Elife*, vol. 6, Jul. 2017, doi: 10.7554/ELIFE.18247.

[82] F. Faul, E. Erdfelder, A. G. Lang, and A. Buchner, "G*Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences," *Behav Res Methods*, vol. 39, no. 2, pp. 175–191, 2007, doi: 10.3758/BF03193146/METRICS.

[83] Student, “The Probable Error of a Mean,” *Biometrika*, vol. 6, no. 1, p. 1, Mar. 1908, doi: 10.2307/2331554.
[84] L. A. Schwarz and L. Luo, “Organization of the locus coeruleus-norepinephrine system,” *Curr Biol*, vol. 25, no. 21, pp. R1051–R1056, 2015, doi: 10.1016/J.CUB.2015.09.039.
[85] D. Seo *et al.*, “A locus coeruleus to dentate gyrus noradrenergic circuit modulates aversive contextual processing,” *Neuron*, vol. 109, no. 13, pp. 2116-2130.e6, Jul. 2021, doi: 10.1016/J.NEURON.2021.05.006.
[86] B. M. Sharp, “Basolateral amygdala and stress-induced hyperexcitability affect motivated behaviors and addiction,” *Translational Psychiatry 2017 7:8*, vol. 7, no. 8, pp. e1194–e1194, Aug. 2017, doi: 10.1038/tp.2017.161.
[87] T. F. Giustino, K. R. Ramanathan, M. S. Totty, O. W. Miles, and X. Stephen Maren, “Locus Coeruleus Norepinephrine Drives Stress-Induced Increases in Basolateral Amygdala Firing and Impairs Extinction Learning,” *Journal of Neuroscience*, vol. 40, no. 4, pp. 907–916, Jan. 2020, doi: 10.1523/JNEUROSCI.1092-19.2019.
[88] P. H. Janak and K. M. Tye, “From circuits to behaviour in the amygdala,” *Nature 2015 517:7534*, vol. 517, no. 7534, pp. 284–292, Jan. 2015, doi: 10.1038/nature14188.
[89] K. M. Gothard, “Multidimensional processing in the amygdala,” *Nature Reviews Neuroscience 2020 21:10*, vol. 21, no. 10, pp. 565–575, Aug. 2020, doi: 10.1038/s41583-020-0350-y.
[90] Y. Yang and J. Z. Wang, “From Structure to Behavior in Basolateral Amygdala-Hippocampus Circuits,” *Front Neural Circuits*, vol. 11, Oct. 2017, doi: 10.3389/FNCIR.2017.00086.
[91] A. Vazdarjanova and S. Maren, “Does the basolateral amygdala store memories for emotional events?,” *Trends Neurosci*, vol. 23, no. 8, p. 345, Aug. 2000, doi: 10.1016/S0166-2236(00)01599-X.
[92] J. G. McCall *et al.*, “Locus coeruleus to basolateral amygdala noradrenergic projections promote anxiety-like behavior,” *Elife*, vol. 6, Nov. 2017, doi: 10.7554/eLife.18247.
[93] J. K. Shepherd, S. S. Grewal, A. Fletcher, D. J. Bill, and C. T. Dourish, “Behavioural and pharmacological characterisation of the elevated ‘zero-maze’ as an animal model of anxiety,” *Psychopharmacology 1994 116:1*, vol. 116, no. 1, pp. 56–64, Sep. 1994, doi: 10.1007/BF02244871.
[94] L. B. Tucker and J. T. McCabe, “Behavior of male and female C57Bl/6J mice is more consistent with repeated trials in the elevated zero maze than in the elevated plus maze,” *Front Behav Neurosci*, vol. 11, p. 13, Jan. 2017, doi: 10.3389/FNBEH.2017.00013/BIBTEX.
[95] B. Lecorps, H. G. Rödel, and C. Féron, “Assessment of anxiety in open field and elevated plus maze using infrared thermography,” *Physiol Behav*, vol. 157, pp. 209–216, Nov. 2016, doi: 10.1016/j.physbeh.2016.02.014.
[96] M. L. Seibenhener and M. C. Wooten, “Use of the Open Field Maze to Measure Locomotor and Anxiety-like Behavior in Mice,” *J Vis Exp*, no. 96, p. 52434, Feb. 2015, doi: 10.3791/52434.
[97] T. Hainmueller and M. Bartos, “Dentate gyrus circuits for encoding, retrieval and discrimination of episodic memories,” *Nature Reviews Neuroscience 2020 21:3*, vol. 21, no. 3, pp. 153–168, Feb. 2020, doi: 10.1038/s41583-019-0260-z.

[98] M. A. Kheirbek *et al.*, “Differential control of learning and anxiety along the dorsoventral axis of the dentate gyrus,” *Neuron*, vol. 77, no. 5, pp. 955–968, 2013, doi: 10.1016/j.neuron.2012.12.038.
[99] B. E. Bernier *et al.*, “Dentate gyrus contributes to retrieval as well as encoding: Evidence from context fear conditioning, recall, and extinction,” *Journal of Neuroscience*, vol. 37, no. 26, pp. 6359–6371, 2017, doi: 10.1523/JNEUROSCI.3029-16.2017.
[100] B. E. Bernier *et al.*, “Dentate gyrus contributes to retrieval as well as encoding: Evidence from context fear conditioning, recall, and extinction,” *Journal of Neuroscience*, vol. 37, no. 26, pp. 6359–6371, 2017, doi: 10.1523/JNEUROSCI.3029-16.2017.
[101] X. Liu *et al.*, “Optogenetic stimulation of a hippocampal engram activates fear memory recall,” *Nature*, vol. 484, no. 7394, pp. 381–385, Apr. 2012, doi: 10.1038/nature11028.
[102] T. F. Giustino and S. Maren, “Noradrenergic modulation of fear conditioning and extinction,” *Frontiers in Behavioral Neuroscience*, vol. 12. Frontiers Media S.A., Mar. 13, 2018. doi: 10.3389/fnbeh.2018.00043.
[103] M. A. Kheirbek, K. C. Klemenhagen, A. Sahay, and R. Hen, “Neurogenesis and generalization: a new approach to stratify and treat anxiety disorders,” *Nature Neuroscience 2012 15:12*, vol. 15, no. 12, pp. 1613–1620, Nov. 2012, doi: 10.1038/nn.3262.
[104] N. L. Balderston, A. Mathur, J. Adu-Brimpong, E. A. Hale, M. Ernst, and C. Grillon, “Effect of anxiety on behavioural pattern separation in humans,” *http://dx.doi.org/10.1080/02699931.2015.1096235*, vol. 31, no. 2, pp. 238–248, Feb. 2015, doi: 10.1080/02699931.2015.1096235.
[105] J. B. Aimone, W. Deng, and F. H. Gage, “Resolving New Memories: A Critical Look at the Dentate Gyrus, Adult Neurogenesis, and Pattern Separation,” *Neuron*, vol. 70, no. 4, pp. 589–596, May 2011, doi: 10.1016/J.NEURON.2011.05.010.
[106] E. E. Bernstein and R. J. McNally, “Exploring behavioral pattern separation and risk for emotional disorders,” *J Anxiety Disord*, vol. 59, pp. 27–33, Oct. 2018, doi: 10.1016/J.JANXDIS.2018.08.006.
[107] J. M. P. Baas, “Individual differences in predicting aversive events and modulating contextual anxiety in a context and cue conditioning paradigm,” *Biol Psychol*, vol. 92, no. 1, pp. 17–25, Jan. 2013, doi: 10.1016/J.BIOPSYCHO.2012.02.001.
[108] C. Grillon, “Startle reactivity and anxiety disorders: aversive conditioning, context, and neurobiology,” *Biol Psychiatry*, vol. 52, no. 10, pp. 958–975, Nov. 2002, doi: 10.1016/S0006-3223(02)01665-7.
[109] J. E. LeDoux, “Emotion Circuits in the Brain,” *Annu Rev Neurosci*, vol. 23, no. 1, pp. 155–184, Mar. 2000, doi: 10.1146/annurev.neuro.23.1.155.
[110] S. Maren, “Neurobiology of Pavlovian Fear Conditioning,” *Annu Rev Neurosci*, vol. 24, no. 1, pp. 897–931, Mar. 2001, doi: 10.1146/annurev.neuro.24.1.897.
[111] A. M. Zador, “A critique of pure learning and what artificial neural networks can learn from animal brains,” *Nature Communications 2019 10:1*, vol. 10, no. 1, pp. 1–7, Aug. 2019, doi: 10.1038/s41467-019-11786-6.
[112] B. A. Richards *et al.*, “A deep learning framework for neuroscience,” *Nat Neurosci*, vol. 22, no. 11, pp. 1761–1770, Nov. 2019, doi: 10.1038/S41593-019-0520-2.
[113] G. R. Yang and X. J. Wang, “Artificial Neural Networks for Neuroscientists: A Primer,” *Neuron*, vol. 107, no. 6, pp. 1048–1070, Sep. 2020, doi: 10.1016/J.NEURON.2020.09.005.

[114] F. H. Sinz, X. Pitkow, J. Reimer, M. Bethge, and A. S. Tolias, “Engineering a Less Artificial Intelligence,” *Neuron*, vol. 103, no. 6, pp. 967–979, Sep. 2019, doi: 10.1016/J.NEURON.2019.08.034.
[115] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/NECO.1997.9.8.1735.
[116] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” *arXiv,* Dec. 2014, doi: https://arxiv.org/abs/1412.3555v1
[117] C. Pandarinath *et al.*, “Inferring single-trial neural population dynamics using sequential auto-encoders,” *Nat Methods*, vol. 15, no. 10, pp. 805–815, Oct. 2018, doi: 10.1038/S41592-018-0109-9.
[118] L. Y. Prince, S. Bakhtiari, C. J. Gillon, and B. A. Richards, “CaLFADS: latent factor analysis of dynamical systems in calcium imaging data,” *ICLR 2021 Conference,* Sept. 28, 2020.
[119] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” 2019, doi: 10.5555/3454287.3455008.
[120] D. B. Ehrlich, J. T. Stone, D. Brandfonbrener, A. Atanasov, and J. D. Murray, “PsychRNN: An Accessible and Flexible Python Package for Training Recurrent Neural Network Models on Cognitive Tasks,” *eNeuro*, vol. 8, no. 1, pp. 1–11, Jan. 2021, doi: 10.1523/ENEURO.0427-20.2020.
[121] M. G. Perich *et al.*, “Inferring brain-wide interactions using data-constrained recurrent neural network models”, doi: 10.1101/2020.12.18.423348.
[122] U. Braun, A. Schaefer, R. F. Betzel, H. Tost, A. Meyer-Lindenberg, and D. S. Bassett, “From Maps to Multi-dimensional Network Mechanisms of Mental Disorders,” *Neuron*, vol. 97, no. 1, pp. 14–31, Jan. 2018, doi: 10.1016/J.NEURON.2017.11.007.
[123] M. D. Humphries, “Dynamical networks: Finding, measuring, and tracking neural population activity using network science.,” *Netw Neurosci*, vol. 1, no. 4, pp. 324–338, 2018, doi: 10.1162/NETN_a_00020.
[124] D. P. Kingma and J. L. Ba, “Adam: A Method for Stochastic Optimization,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Dec. 22, 2014, doi: arXiv:1412.6980
[125] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature 1986 323:6088*, vol. 323, no. 6088, pp. 533–536, 1986, doi: 10.1038/323533a0.
[126] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, “On the Properties of Neural Machine Translation: Encoder-Decoder Approaches,” *Proceedings of SSST 2014 - 8th Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, Sep. 2014, doi: 10.48550/arxiv.1409.1259.
[127] J. Humplik and G. Tkačik, “Probabilistic models for neural populations that naturally capture global coupling and criticality,” *PLoS Comput Biol*, vol. 13, no. 9, p. e1005763, Sep. 2017, doi: 10.1371/journal.pcbi.1005763.
[128] M. Nolan, B. Pesaran, E. Shlizerman, and A. Orsborn, “Multi-block RNN Autoencoders Enable Broadband ECoG Signal Reconstruction,” *bioRxiv*, p. 2022.09.07.507004, Sep. 2022, doi: 10.1101/2022.09.07.507004.

APPENDIX A: SUPPLEMENTARY MATERIALS

CaGraph preprocessing Jupyter notebook:

https://github.com/vporubsky/CaGraph/blob/main/preprocessing_tutorial.ipynb

CaGraph visualization Jupyter notebook:

https://github.com/vporubsky/CaGraph/blob/main/visualization_tutorial.ipynb

CaGraph analysis Jupyter notebook:

https://github.com/vporubsky/CaGraph/blob/main/analysis_tutorial.ipynb