Using a Brain Network Model to Predict Brain Structure from Function

by

Nicholas Green

Dr. Shella Keilholz
Adviser

Department of Biology

2019

Using a Brain Network Model to Predict Brain Structure from Function

By

Nicholas Green

Shella Keilholz

Adviser

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Biology

2019

Abstract

Using a Brain Network Model to Predict Brain Structural Connectivity

By Nicholas Green

Brain network models have become a promising theoretical framework in simulating signals that are representative of whole brain activity such as resting state functional-MRI (rs-fMRI). Brain network models use the structural connectivity of the brain as an input and, using a set of differential equations to describe the dynamics, they can be used to simulate brain activity starting from a randomly initialized signal. Previous studies have shown that these models can simulate brain activity that are functionally coordinated as measured through averaged functional connectivity with a high degree of similarity to empirical rs-fMRI data. In this study, we address the inverse problem: by utilizing measured rs-fMRI and treating brain structure as a variable, we formulate and solve the constrained convex optimization problem for the sparsest structural connectivity that, for a given set of dynamics, explains the measured brain activity. We solve the constraint problem with machine learning, using gradient descent on our loss function. In this experiment, three different models are used in order to explore what level of complexity in the model constraints leads to the more accurate solution. These models are based around intuition about the brain and grew in complexity, first penalizing large weights, then penalizing large weights and large tract lengths, and finally using these penalties but with two weight matrices, one being a term that synchronizes brain regions and the other being a term that pushes them apart. We found that the more complex the model, the better the solution, with higher correlation with actual structural connectivity. When only considering within-hemispheric connections, since connections between hemispheres are not as well known, correlations were even higher, going from 0.353 to 0.437 to 0.569 as the model

complexity increased. In previous studies that worked on similar generative models, the simulated connectomes had correlations in the range of 0.3 to 0.5. So, the solutions generated here are quite promising. This experiment shows that we can predict structure from function just as well as function from structure using brain network models. These results open up new avenues to explore with brain network models and possible new models to implement.

Using a Brain Network Model to Predict Brain Structure from Function

By

Nicholas Green

Dr. Shella Keilholz

Adviser

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Department of Biology

2019

Table of Contents

Figures

Introduction

Whole brain activity is complex but critical for understanding behavior and function of the central nervous system. This coordinated activity is thought to be related to the underlying structure or connectome and has driven much interest and excitement in the past decade. Interest in this field has been especially increased since 2009, when the Human Connectome Project (HCP) was launched by the National Institutes of Health with the goal of building a map of all neural connections in the brain and relating it to the measured the functional activity of the cortex (Van Essen, et al, 2013). Understanding how structure relates to function has also shed more light on brain diseases such as Parkinson's (Kamagata, et al., 2017, Wolters, et al., 2018) and Alzheimer's (Schouten, et al., 2017), among many other disorders.

One of these tools to measure whole brain activity is functional magnetic resonance imaging (fMRI). fMRI measures the blood-oxygen-level dependent (BOLD) signal, which is correlated with neural activity. Seiji Ogawa et al (1990) showed that the magnetic properties of the deoxygenated form of hemoglobin is different than that of the oxygenated form. When neurons fire, blood flow is increased to these regions to meet the neurons' energy need. This increased blood flow also brings with it oxygen in the form of oxygenated hemoglobin. The oxygen is usually brought in at a greater rate than oxygen is consumed in burning glucose, causing a net increase in the oxygenated form of hemoglobin. This difference is detectable using MRI scanning. This technique is becoming increasingly popular in the research and in the clinical world, as it is non-invasive and has an excellent spatial resolution (up to a few millimeters) although has a lower temporal resolution (<0.1 Hz). However, the method is not perfect in measuring neural activity, as the BOLD signal often contains significant noise from other sources, such as heart rate and respiratory fluctuations. This provides difficulties in mapping precise brain functional connectivity from the raw fMRI signal.

Resting state fMRI (rs-fMRI) is a specific type of fMRI that is measured with the subject in a resting state, which simply means that during the scanning, the patient is not told to perform any explicit task. rs-fMRI studies have shown that even at rest, there exist complex spontaneous dynamics with intermittent spatiotemporal fluctuation patterns. In fact, further studies have demonstrated that these fluctuations in the BOLD signal correlate across functionally related brain regions (Bharat Biswal et al, 1995; Hutchison et al, 2013). Functional connectivity (FC) refers to the connectivity between brain regions of interest (ROIs) that activate together for a particular task or during rest and is measured by correlating these regions over long periods of time (Smith 2009).

In recent years, analysis of FC data has moved beyond interpreting average statistical relationships over the course of a long scan to dynamic analysis methods that analyze the coordination of brain activity on a moment to moment basis (Shakil et al., 2016). The anatomical connections of the brain are assumed to remain constant, so that the time-dependent coordinated activity plays out over the same framework of structural connectivity (SC) based on white matter connections over time (Cabral et al., 2017; Deco et al., 2017; Shen et al., 2015).

Structural connectivity can be measured by a variety of techniques, including diffusion MRI. Diffusion MRI (DWI) uses the fact that the diffusion of water molecules in tissues reflects interactions with obstacles such as macromolecules, fibers, and membranes. This diffusion is measured by applying strong gradients and then measuring how easily water can move along a certain area; the less ease of movement there is, the stronger signal will be. And, because tightly packed bundles of axons in the interior of the brain hinder water movement perpendicular to themselves, these bundles, known as white matter, can be detected and visualized in great detail. These diffusion patterns can thus reveal microscopic details about tissue architecture (Basser, et al, 2000, Bihan, 2014). For analysis, the brain is often separated into distinct anatomical regions corresponding to functional divisions within the brain, like the precentral gyrus or lingual gyrus (see Table 1 for a full list of the parcellations used), and the connectivity between those regions can be estimated using tractography. In tractography, fiber tracking algorithms can be used to track fibers along their whole lengths and thus build up an estimate of SC in the whole brain. This information can be converted into a weighted adjacency matrix, called a structural connectome.

To relate structure to function, we can model the brain's activity as interactions of these ROIs, connected by a structural network. The change in activity of a certain region is modeled as a function of its own local state, as well as the activity of its neighboring regions, which can either excite or inhibit that region's activity. This influence between regions is thought to be positively dependent on the strength of the connections between the regions; regions that have lots of connections will have lots of influence on each other, and less so for less connected regions. The nature of this relationship can be studied by mathematical models of brain activity, known as brain network models (BNMs) or neural mass models, that attempt to simulate this signal,

Therefore, in a BNM, the activity of each ROI can be modelled as a function of its local processing state plus a delayed function of the activity of its network neighbors (Breakspear, 2017; Sanz Leon et al., 2015). Each ROI is connected via a weight matrix obtained from structural connectivity that describes the strength of the connection between nodes. In general each of these n areas are modeled by a differential equation for each node: $\frac{d\,r(t)}{dt} = f(\mathrm{r}(t), W, L)$ where r(t) is the timeseries of all the ROIs, W is the weight matrix, L is the length matrix, and for given random initial conditions for $n0$, the timeseries n(t) can be solved for by using the Euler integration method (Sanz et al., Leon 2015). r(t) is the state variable which represents a measurable property of the neural mass such as firing rate.

One such model is the Firing Rate model (also called the General Linear Model in literature). This model has been shown to be robust in reproducing brain activity and has a relatively small number of parameters to optimize (Deco et al., 2010; Cabral et al., 2017). The equation for the model is given by: $\tau_0 \frac{dr_n}{dt} = -r_n(t) + \frac{k}{c_1}\sum_{p=1}^{N} C_{np} r_p\left(t - \tau_{np}\right) + \sigma n(t)$, where r(t) represents the mean firing rate, $C_{np}$is the weight between nodes n and p, τ_{np} is the delay time between nodes n and p, σn is the standard deviation of gaussian white noise, c_1 is the first eigenvalue of the weight matrix C, k is the coupling parameter, and τ_0 is the relaxation constant. Inputs into this model from neighboring brain regions shift the mean firing rate to a higher firing rate. The mass shifted from its equilibrium tries to relax at the rate proportional to its own firing rate, keeping the system stable via negative feedback.

This model has been shown in the past to model real brain dynamics, such as average FC, relatively successfully. In a previous study, average functional connectivity matrices were estimated using correlation pairwise across all ROIs (Figure 1) (Kashyap et al., 2018). To quantify the similarity between the simulated FC matrices and the empirical FC matrix, the correlation between the two was calculated, a method used in other previous studies (Cabral et al., 2011). Correlation was 0.5 between the Firing Rate model and rs-fMRI FC matrices. These are in the range of values reported in earlier literature in other BNMs [0.3, 0.7] (Senden et al., 2017; Cabral et al., 2011).

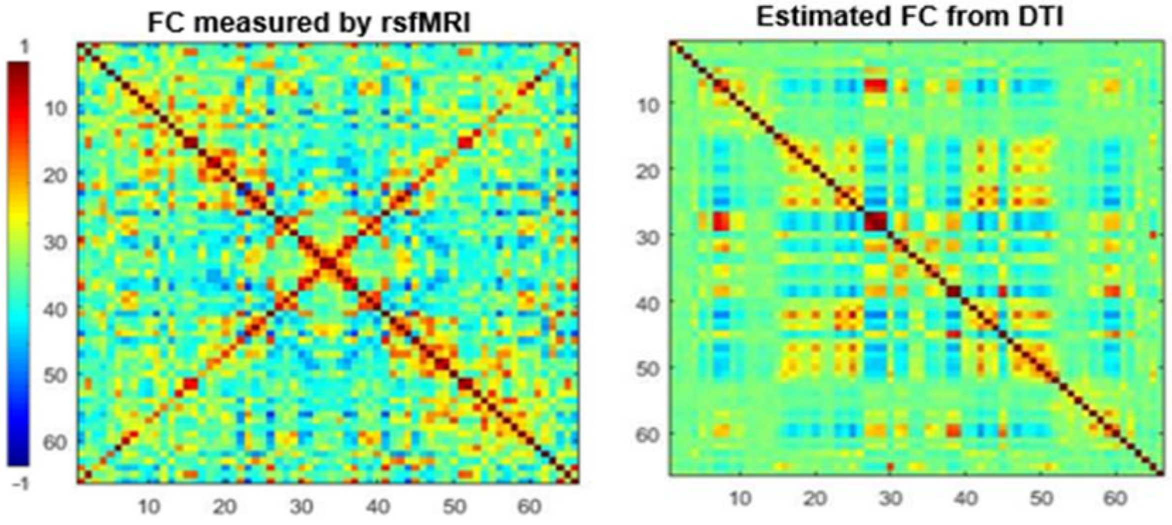


Fig 1: Comparison of the average functional connectivity between the rsfMRI signals and a simulated model. Correlation between the matrix for empirical data and the Firing Rate model is 0.5. The modeled matrix and the empirical data exhibit similar structure such as the coordination between hemispheres. Antidiagonal corresponds to matching areas on opposite hemispheres, which is strong in the measured FC, but not in the estimated model. (Kashyap et al., 2018)

In this figure, and in all following figures displaying functional and structural connectivity matrices, each axis represents the 66 ROIs in our parcellations, 1-33 for one hemisphere, and 34-66 for the other. Therefore, the top left and bottom right quadrants represent connections within hemispheres, and the top right and bottom left quadrants represent between-hemisphere connections.

We wish to extend these results in relating structure to function, by utilizing these BNMs and measured rs-fMRI activity in order to estimate the underlying structural connectome. Our goal is to use machine learning techniques to estimate a structural connectivity and plug it into the BNM in order to predict the change in rs-fMRI signal. The loss function is then defined as the difference between the estimated change in rs-fMRI and the actual change. Our results will not only strengthen our understanding between SC and FC, but will also allow us to estimate the

SC of regions of the brain that are known to have problems in diffusion imaging such as the long range inter-hemispheric connections. Furthermore, we hypothesize that the structural connectivity of the brain is constrained by its resources. The brain is bounded by the skull, and thus cannot produce an infinite number of neuronal tracts; it must use its space in an efficient way. The most efficient solution would be to produce the minimum number of tracts of neurons possible to still satisfy the needed functional connectivity. Thus, we predict that this minimum solution for a given FC will precisely be the structural connectivity. In other words, we hypothesize that the most efficient way for the brain to generate its functional connectivity is through the actual measured SC. This hypothesis fits with our brain network model. The connectome in the model constrains the contribution by the surrounding regions to each region's firing rate. Therefore, if we find a sparse weight matrix that satisfies the BNM, given the firing rate data, we hypothesize that it will closely resemble the actual structural connectivity of the brain.

Methods

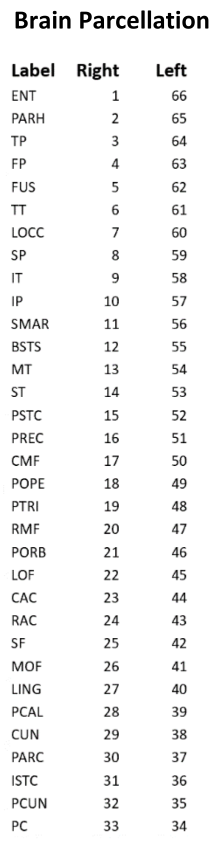
Data

Data was taken from the MyConnectome project, by Dr. Russ Poldrack, Department of Psychology, Stanford University. The project seeks to characterize brain function and metabolism in a single individual over the course of one year. This dataset is well suited for this experiment because having all of the data, structural and functional, from one individual will allow for precise knowledge of the accuracy of the model – one can directly compare the predicted structural connectome generated using rs-fMRI data to the actual structural connectome in that same individual. This makes our results robust by circumventing the problem of individual differences. The rs-fMRI data will serve as the input to our model, and the structural data will be used to compare to the model results. Taking the average of all of the structural connectomes we generate will serve as a strong and accurate measure of ground truth. Structural connectivity is the variable of interest here, and having this ground truth will allow us to know exactly how accurate our estimated SC is. For this experiment,

- 84 rs-fMRI runs (TR=1.16 ms, TE = 30 ms, flip angle = 63 degrees, voxel size = 2.4 mm X 2.4 mm X 2 mm, distance factor=20%, 68 slices, oriented 30 degrees back from AC/PC, 96×96 matrix, 230 mm FOV, MB factor=4, 10:00 scan length),
- 15 diffusion weighted scans (b=1000/2000, 1.74 X 1.74 X 17 mm resolution, 72 slices, FOV=223 mm, 128 X 128 matrix, TR=5000 ms, TE = 108 ms, PAT=2, MB factor=3),
- and 1 T1-weighted image (saggital, 256 slices, 0.7 mm isotropic resolution, TE=2.14 ms, TR=2400 ms, TI=1000 ms, flip angle = 8 degrees, PAT=2, 7:40 scan time)

were used (see Fig 3a, b for raw T1, diffusion image).

Tractography



Brain Parcellation

Label	Right	Left
ENT	1	66
PARH	2	65
TP	3	64
FP	4	63
FUS	5	62
TT	6	61
LOCC	7	60
SP	8	59
IT	9	58
IP	10	57
SMAR	11	56
BSTS	12	55
MT	13	54
ST	14	53
PSTC	15	52
PREC	16	51
CMF	17	50
POPE	18	49
PTRI	19	48
RMF	20	47
PORB	21	46
LOF	22	45
CAC	23	44
RAC	24	43
SF	25	42
MOF	26	41
LING	27	40
PCAL	28	39
CUN	29	38
PARC	30	37
ISTC	31	36
PCUN	32	35
PC	33	34

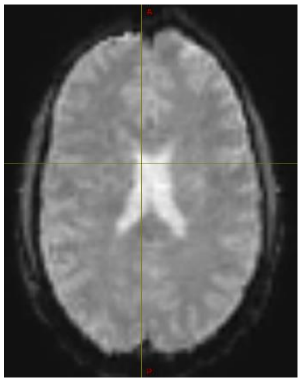
Figure 2: parcellation of the brain used in these experiments, from Desikan et al., 2006: 66 regions, 33 per hemisphere.

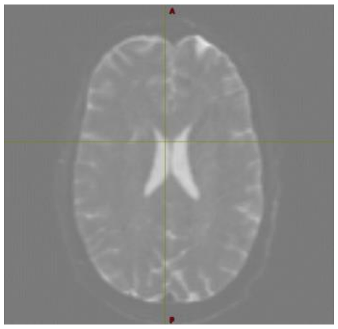
Abbreviations: ENT = entorhinal, PARH = parahippocampal, TP = temporal pole, FP = frontal pole, FUS = fusiform, TT = transverse temporal, LOCC = lateral occipital, SP = superior parietal, IT = inferior temporal, SMAR = supramarginal, BSTS = banks of the superior temporal sulcus, MT = middle temporal, ST = superior temporal, PSTC = postcentral, PREC = precentral, CMF = caudal middle frontal, POPE = pars opercularis, PTRI = pars triangularis, RMF = rostral middle frontal, PORB = pars orbitalis, LOF = lateral orbitofrontal, CAC = caudal anterior cingulate, RAC = rostral anterior cingulate, SF = superior frontal, MOF = middle orbitofrontal, LING = lingual, PCAL = pericalcarine, CUN = cuneus, PARC = paracentral, ISTC = isthmus cingulate, PCUN = precuneus, PC = posterior cingulate

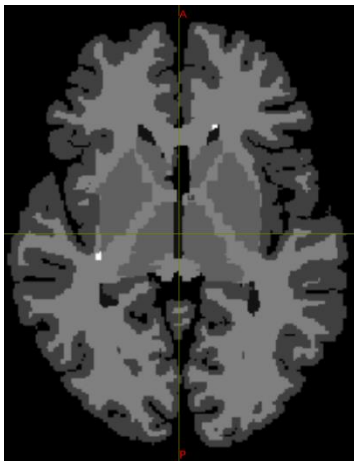
To generate the structural connectomes, tractography was performed using MRtrix, an open source software used for diffusion MRI analysis. First, the T1 and diffusion images were skull stripped and aligned to each other in the standard MNI space using FSL, another software library for MRI analysis. Then, the diffusion image was preprocessed, denoised, and biascorrected using MRtrix. The response function, the signal expected for a voxel containing a single, coherently oriented bundle of axons, was estimated using the tournier algorithm in MRtrix. Using this, constrained spherical deconvolution was performed, which estimates a white matter fiber orientation distribution (FOD) function, which is used in the tractography algorithm. Using the T1 image, a tissue segmented image was generated, with labels corresponding to the various ROIs (Fig 3c). Using this image and the FOD image, automatically constrained tractography was performed, generating 10 million streamlines. Then, an algorithm in MRtrix was used to assign weights to each of these tracts. Using this estimate and the parcellation image generated from using the Desikan-Killiany atlas (Desikan et al., 2006), a structural connectome was generated, scaling each value by the inverse of the sum of the volumes of the two nodes (Fig 3d). This scaling is done because a larger node is more likely to be intersected by any particular streamline, so scaling by the inverse of the node volumes attempts to directly compensate for this effect. A length matrix was also generated by following this same method, but scaling by the length of the two nodes instead of the inverse of the volumes. This matrix will have different structure than the weight matrix because those connections that have the longest lengths, such as the connections between hemispheres, do not necessarily have the most connections between them. The weight matrix will serve as the ground truth to compare with our models, but having a length matrix is also useful for one of the models detailed below. In brief, it provides additional information about the brain that we can include in the model to better constrain the weights we are solving for. The connectomes were then reordered so that the indices corresponded with the 66 ROIs given in Table 1 (Fig 3e, f, g). Tractography was performed on 15 diffusion images in total, creating 15 connectomes, which were then averaged in order to have a robust estimate of the ground truth, since a single diffusion image contains noise and may result in a slightly inaccurate connectome.

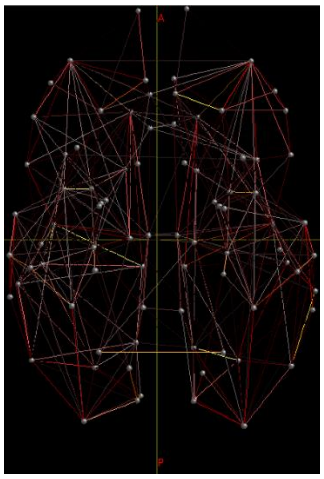
rs-fMRI Preprocessing

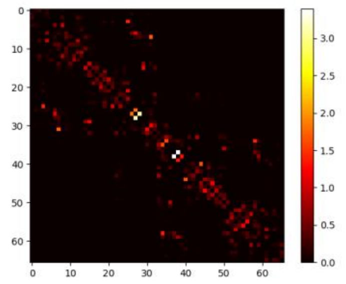
The rs-fMRI data was preprocessed, starting from the raw timeseries surface data. The data was ICA denoised using the 300 ICA vectors that HCP provides. We then applied the same Desikan-Killiany atlas as used in the tractography onto the data and obtained the mean time series for each ROI. The next steps in the preprocessing pipeline were z-scoring each time series, then band passing filtering the signal from 0.01 to 0.25 Hz, and then global signal regression using a linear regression model, and then applying a final z-score step. These steps were selected in accordance with Cabral et al. (2011). This is the data that will be used as the input to the brain network model. The preprocessed data has the structure of a 66x84x518 matrix: it is the BOLD activity of 518 seconds of each of the 66 regions of interest in our parcellation, repeated for 84 scans total over the course of the MyConnectome data collection period. However, it is also useful to visualize functional connectivity. To do this, the correlation coefficient between each region was calculated to generate the functional connectivity matrix shown in Figure 4. This matrix is very similar to the FC shown in Figure 1, which was measured from another dataset, with perhaps slightly more noise. Because there were no significant differences when comparing to this other data set, we assumed that these data are generally reliable.

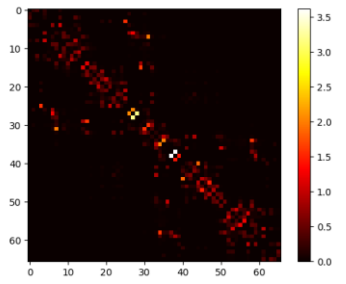
a)


b)


c)


d)


e)


f)


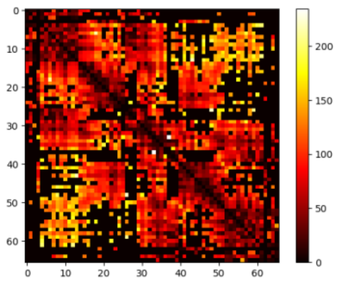
g)


Fig 3: Various stages in tractography. a) raw T1 weighted image. b) raw diffusion weighted image, c) tissue segmented image, d) structural connectome, with the various tracts visualized, e) single structural connectome, scaled by inverse of volume in matrix form, f) average of 15 structural connectomes, g) structural connectome scaled by length, in matrix form.

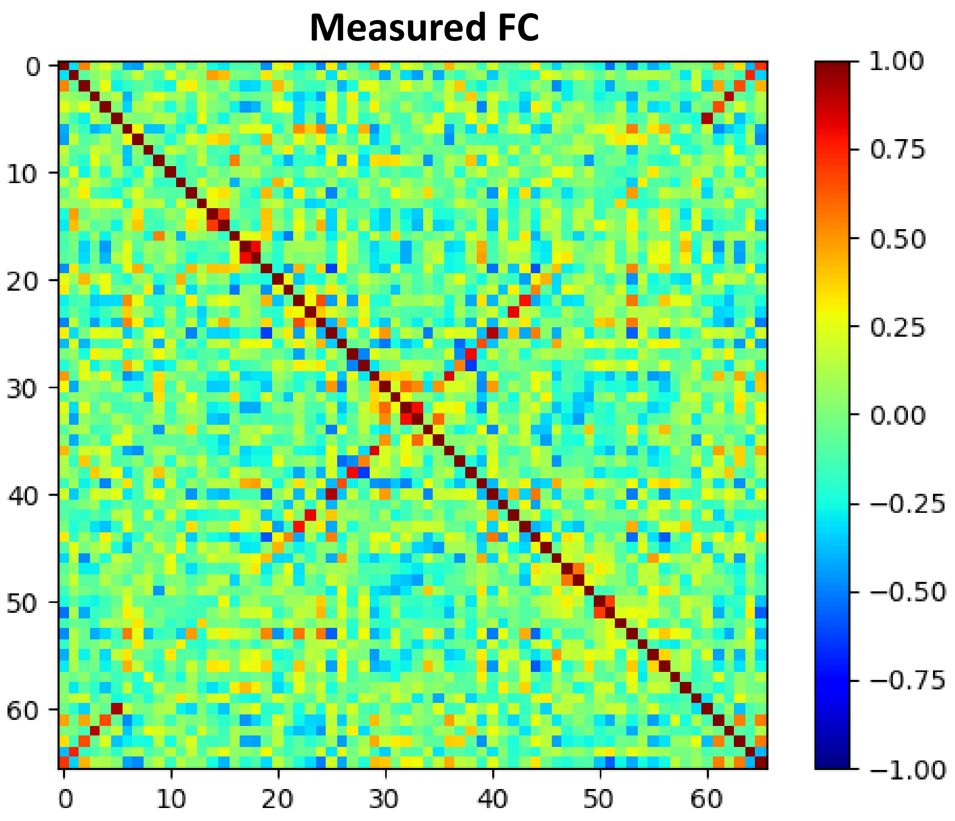


Fig 4: Average functional connectivity matrix. Generated by taking correlation between each region in parcellated rs-fMRI timeseries data, average of all 84 runs.

Brain Network Model Equation

For this experiment, a simplified brain network model equation was used:

$$\overline{r_n(t)} = -r_n(t) + |C - I| * r_p(t)$$

Where $r_n(t)$ is the firing rate for region n, C is the weight matrix, I is the identity matrix, and $r_p(t)$ is the firing rate for surrounding regions p. We make the diagonal equal to zero in this model because an ROI cannot be connected to itself. Then, the absolute value is taken to enforce positive weights, since ROIs cannot have negative structural connectivity; they are either connected, and thus would have a positive weight, or not connected, and would have a zero weight. This simplified model still preserves the fundamental components of the BNM; the exponential decay term and the weighted contribution from the other regions of interest. However, this model fits better with the optimization algorithms discussed shortly. Through this simplification, however, we lose a couple terms that were present in the more complex Firing Rate model: the scaling by a factor of the first eigenvalue of the weight matrix, and the time delay of r_p. The first factor can be added back in relatively easily in future experiments if the simple model works well, and its omission will not affect the correlation between the solution given from this simple model and the solution given by the Firing Rate model. The time delay, however, is difficult to represent in our neural network model, since neural networks don't keep track of weights from previous timesteps. Therefore, in this model, all interactions between ROIs are represented as instantaneous. If a way is found to represent this more easily, it will be included in future studies.

Of course, the solution that minimizes the loss using this simpler equation will scaled to a different order of magnitude than that from the original BNM equation, as this new equation is missing the scalars detailed above that are present in the original, but the results of this model can still be interpreted by examining the correlation between the weight matrix we solve for and the actual structural connectome.

Optimization Method

Convex optimization is a technique that seeks to find the minimum value of a convex function. In our case, the function we are trying to minimize is the loss between the estimated change in activity generated from the model and the actual, measured change from the rs-fMRI data. Each iteration of the model, using gradient descent, we update the weight matrix to ‘walk’ in the direction of the steepest rate of change of the loss function, in order to move closer to the minimum value. When the model converges on a weight matrix that minimizes the loss, we compare that matrix to the actual SC.

There are many optimization techniques of varying complexity available using built-in software libraries, but for this experiment, we used machine learning to solve the constraints using regression. Because we are doing gradient descent to find an optimal solution, a more complicated, convex optimizer is not necessary for this stage in the experiment, although it would be interesting to test in the future to see how performance changes (see Future Directions). In this model, we first randomly choose one of the 84 rs-fMRI runs to input into the model. The weight matrix, which we are solving for, is initialized randomly (uniformly over the interval [0, 1)).

These data are all plugged into the simplified brain network model equation detailed above to obtain an estimate for the change in BOLD signal with respect to time. This is compared to the actual change over time, calculated by taking the input timeseries, subtracting each timepoint from the next timepoint in the series, and diving by the sampling time (1.16 seconds in this case). The error between the actual and estimated derivatives is measured using mean squared error ($MSE = \frac{1}{N}\sum_{i=1}^{N}(Actual_i - Estimated_i)^2$). This loss was backpropagated to the BMN equation using the chain rule to find the derivative of the error with respect to C, the weight matrix: $\frac{d\,MSE}{d\,C} = \frac{d\,MSE}{d\,Estimated} * \frac{d\,Estimated}{d\,C}$. Then, the matrix C is updated using gradient descent, making the updated weights equal to the current weights minus this derivative, times some learning rate. This process was repeated, using a randomly selected input from the 84 each time, for 15,000 iterations. Tensorflow, an open-source, machine learning library for the Python programming language, was used to automate and quicken much of the above process (see Appendix for Python code).

Regularization and Length Penalty

Because this model mimics a real-life resource constraint problem, it is necessary to promote a sparse weight matrix during the machine learning process outlined above, as without it, the problem is unconstrained and has many solutions. To achieve this, one can give a greater loss to a dense matrix by adding some factor of the norm of the matrix to the loss function. For this Tensorflow's norm function was used, which uses by default the Frobenius norm, defined as the square root of the sum of the absolute squares of its elements. Thus, a large matrix will have a large norm, and since large norms will result in a large loss, sparsity will be encouraged. The penalty term used in this model is $\lambda||C||$, where C is the weight matrix, the double vertical lines represent the norm, and λ is a free parameter. This λ is estimated by manually running the model over several values of λ, and choosing the value that gave the solution with the best correlation with our ground truth connectome (see Analysis section for detail on correlation measures). This model will be called Model 1.

In addition to this model, to model more closely the anatomical tracts of the brain, we also built another model that penalized entries with a longer tract length in addition to penalizing the weight matrix itself, as defined in the length matrix generated by tractography. Since longer tracts between nodes will require more resources in the brain, these longer tracts are discouraged, and thus even more sparsity is encouraged among these entries. Thus, the penalty term used in this model is $\lambda||C * L'||$, where $L' = \gamma * L$, L being the length matrix and γ the scaling factor on the length matrix (a free parameter), with the other variables being as described above. This model will be called Model 2.

Separated Weight Matrices

In addition to the design detailed above, a slightly more complex model was implemented as well in the attempt to better model the actual dynamics of the brain. Some areas in the brain synchronize with each other, strengthening the signal that comes to those nodes, and others push each other apart, diminishing their neighbors' activity. Therefore, representing these two types of networks separately during training could produce results that better resemble the actual SC of the brain. To model this, instead of solving for a single weight matrix, we use two matrices in the model: one representing weights between synchronizing networks and the other representing weights between networks that push apart. For each region in the brain, its firing rate would then depend positively on the neighbors it synchronizes with and negatively on the neighbors it pushes apart, in addition to the usual decay term. The brain network model equation then becomes: $\tau_0 \overline{r_n(t)} = -r_n(t) + C_{pos} * r_p(t) - C_{neg} * r_p(t)$. To obtain the final weight matrix, after training, the two matrices were simply added together. The penalty term then becomes: $\lambda_1||C * L'|| + \lambda_2||C||$, with λ_1 and λ_2 instead of a single λ. C and L' are as described above. This model was tested in addition to the two simpler models described above. This model will be called Model 3.

Analysis

To measure accuracy of the models, we measured the accuracy between the final estimated weight matrix and the averaged structural connectome using the correlation coefficient between the two matrices, because this measure has been used in past studies (Cabral et al., 2011). This measure also works well because the final estimated matrix will be scaled differently than the actual connectome because our brain network model equations were slightly simplified and thus does not include some scalars present in the full equation (see Firing Rate equation above).

Inter-hemispheric connections in the brain are known to be unreliable in DWI imaging because the connections are much sparser than the intra-hemispheric areas. Therefore, analyzing only the intra-hemispheric connections may provide a more useful measurement, so this was done in addition to analysis on the full connectomes. In the connectivity matrix, this would correspond to the top left and bottom right quadrants.

Results

No Model: Initial matrix analysis

For comparison, we calculated the correlations of both a randomly initialized matrix (uniform [0,1)) and a single structural connectome with the averaged structural connectome. For the random matrix, correlation was 0.003 and 0.007 for the intra-hemispheric areas only, and for the single structural connectome, correlation was 0.995 to the averaged structural connectome, for both the full matrix and the intra-hemispheric areas only.

Model Using Single Matrix with Weight Penalty (Model 1)

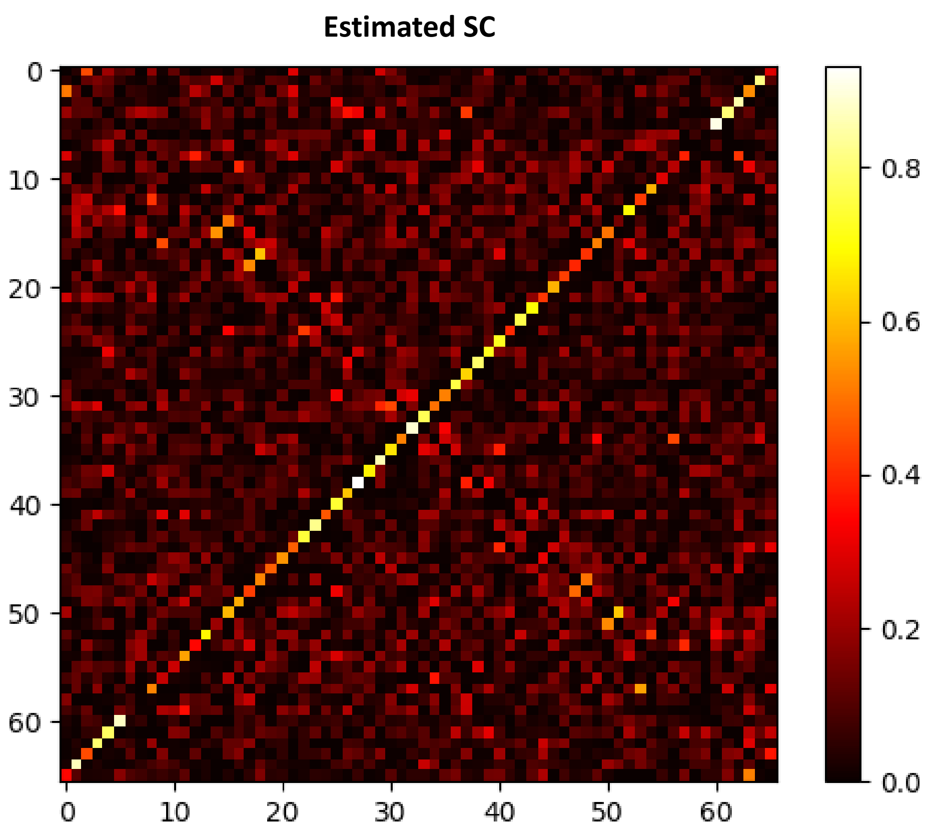


Fig 5: Estimated connectome using single matrix model. 15,000 iterations, penalty = 0.009, learning rate = 0.005. Correlation with averaged measured connectome = 0.223. Correlation of intra-hemispheric regions with averaged measured connectome = 0.353. Matrix randomly initialized (uniform [0, 1)).

Performing gradient descent on the model with the single matrix with a weight penalty only led to the estimated connectomes shown in Fig 5. The estimated connectome using a random initialization has a correlation of 0.223 with the averaged connectome from the measured data. For the intra-hemispheric connections only, correlation was 0.353.

In this simple model, the model already is doing fairly well, having a low to moderate correlation to our ground truth connectome. The solution is much better than the initial correlations with the random matrix, as recorded in the last section. However, there is much noise in the solution, making it difficult to visualize the structure of the matrix. The intense band along the anti-diagonal is also perplexing, as it is not present in the ground truth connectome. This doubtless contributes in part to the significant difference between the total correlation and the intra-hemispheric correlation.

Model Using Single Matrix with Weight and Length Penalty (Model 2)

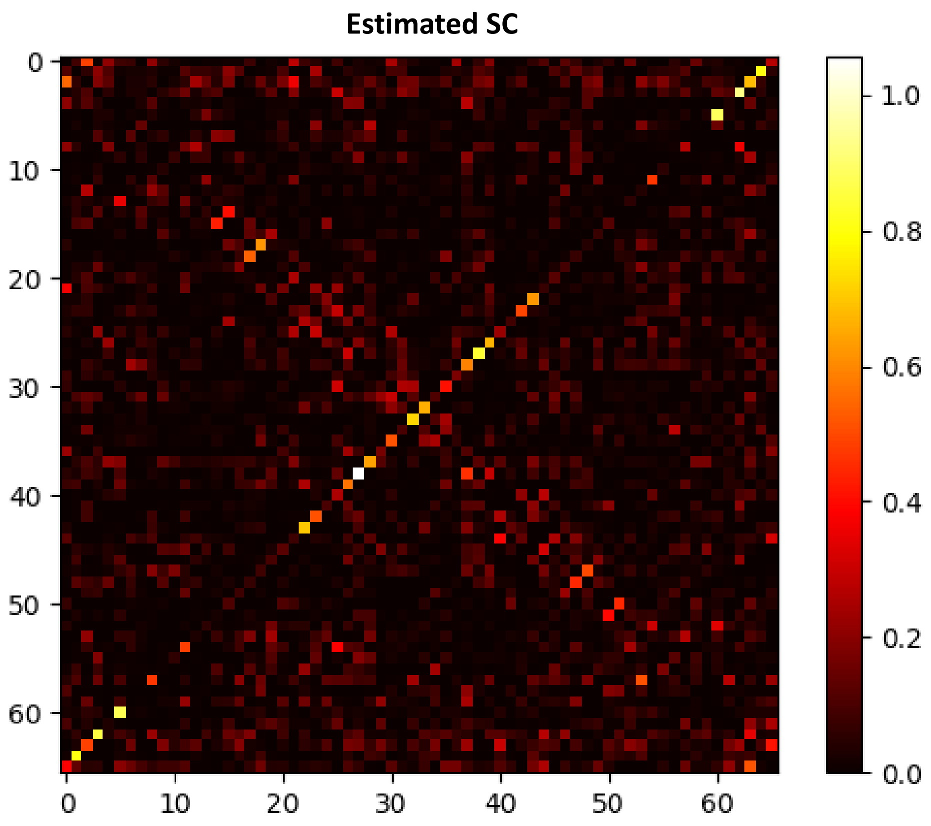


Fig 6: Estimated connectome using single matrix model with added length penalty. 15,000 iterations, weight penalty = 0.009, length factor= 0.1, learning rate = 0.005, Correlation with averaged measured connectome = 0.279. Correlation of intra-hemispheric regions with averaged measured connectome = 0.437. Matrix randomly initialized (uniform [0, 1)).

Gradient descent with the single matrix model with a penalty on the product of the weight and length matrices only led to the estimated connectome shown in Fig 6. For the model initialized with a random matrix, correlation was 0.279, with intra-hemispheric correlation 0.437. The sparsity penalty in this case was 0.009, with the length scaled by 0.1. Learning rate was 0.005.

This randomly initialized matrix model does significantly better at simulating the actual structural connectivity, as the correlation with the actual SC is 0.056 higher than Model 1 and 0.084 higher for the intra-hemispheric areas. The estimated connectome shares much structure with the actual connectome, especially intra-hemispherically. This is most pronounced in the areas surrounding the diagonal, where significant structure can be seen through the noise. However, there is still lots of noise in the model, especially in the inter-hemispheric areas. This makes the total correlation much lower than the correlation of the intra-hemispheric areas.

Model Using Two Matrices with Length Penalty (Model 3)

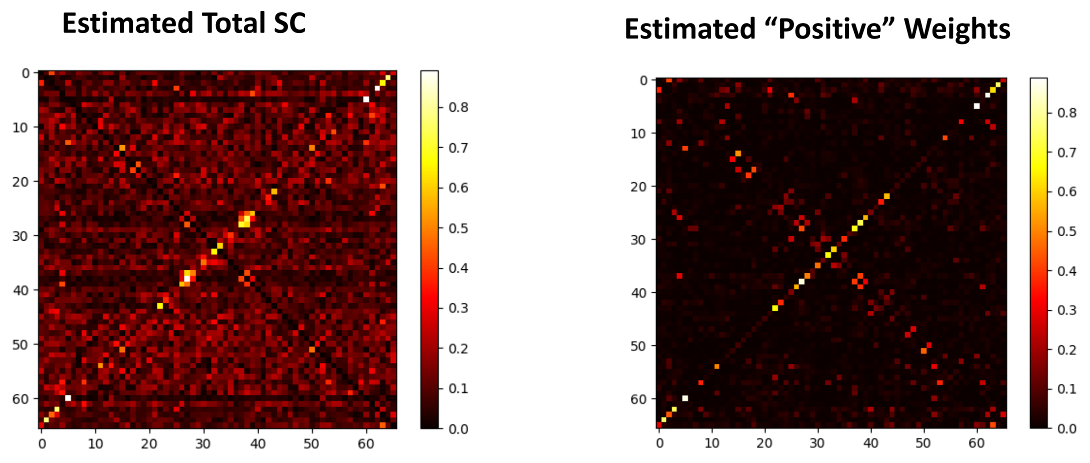


Fig 7: Left: Estimated connectome using separated matrix model with added length penalty. 15,000 iterations, weight penalties = 0.005, 0.015, length factor = 0.1, learning rate = 0.005. Correlation with averaged measured connectome = 0.153. Correlation of intra-hemispheric

regions with averaged measured connectome = 0.284. Matrix randomly initialized (uniform [0, 1)).

Right: Final weight matrix representing positively associated weights. 15,000 iterations, weight penalties = 0.005, 0.015, length factor = 0.1, learning rate = 0.005. Correlation with averaged measured connectome = 0.320. Correlation of intra-hemispheric regions with averaged measured connectome = 0.569. Matrix randomly initialized (uniform [0, 1)).

For model 3, the simulated connectome representing the sum of both weight matrices had a correlation that was remarkedly lower than the other models, with correlation of 0.153 and intra-hemispheric correlation of 0.284 (Fig 7, left). However, when analyzing the two matrices separately, some light can be shed on this result. The correlation of the matrix representing only the positively synchronized edges was 0.320, with intra-hemispheric correlation of 0.569 (Fig 7, right), while the matrix representing negative edges had a correlation of -0.091. This could mean that what we have been calling "negative" edges do not actually exist as real tracts in the brain, since they are not correlated with brain structure in any way, and that the brain ROIs are mainly sending glutamatergic input to each other. But, clearly, including them in the model did improve our accuracy on the positive connections.

Therefore, in these results and in the discussion below, the correlation of the "positive" matrix will be the value reported, since it shows the highest correlation with the actual brain SC. This matrix shows the best result of all of the models, with the highest correlation by far, as well as a significant reduction in noise.

A summary of all results is shown in table 1.

Summary of Results

	Correlation	Intra-hemispheric correlation
Model 1:	0.223	0.353
Model 2:	0.279	0.437
Model 3:	0.320 (pos) 0.153 (sum)	0.569 (pos) 0.284 (sum)

Fig 8: Summary of results of all models, using random initialization using a uniform distribution between 0 and 1. Accuracy measure: Correlation between model solution and average of 15 measured structural connectomes. For Model 3, correlations for both the positively associated edges only and total weight matrix are shown.

Discussion

This exploratory application of convex optimization to empirical rs-fMRI data using a simple BNM produced a variety of interesting data that sheds some light on brain structure and function, especially in terms of the nature of structural constraints on whole brain activity. These data also reveal the limitations of such a model and suggest possibilities for revised models or different directions.

Model Performance

First, we will discuss the performance of the models initialized with a single structural scan. Models 1, 2, and 3 each introduced a new bit of complexity, with Model 3 being the most complex, trying to represent two different types of networks in the brain, as well as adding penalties for the weight and the length. As the complexity of the model increased, the performance of the model got better, with correlation of the intra-hemispheric regions increasing from 0.353 to 0.437 to 0.569 from Model 1 to Model 2 to Model 3. This reflects that fact that brain dynamics are very complex, and thus added complexity in the model, as long as it accurately reflects brain dynamics, will result in a more accurate model. Model 3's intra-hemispheric correlation of 0.569 is a very strong result that provides a very good estimation of structural connectivity with limited noise, especially compared to the other models.

An alternate interpretation of these results could be that the increase in performance was simply due to the increase in parameters, making the model more complex, not because they more accurately represented the brain. In other words, the more complex models could be overfitting the data. However, from Model 1 to Model 2, only one new parameter was introduced (the length penalty), so the increase in performance there is significant compared to the added complexity, which is small. This objection could be valid, though, for Model 3, which essentially doubled the number of parameters with roughly the same increase in performance. Future studies would need to go deeper into these results, using more complex methods of analysis to determine if the better performance is significant given the added complexity.

Since this experiment is the first of its kind in using machine learning on BNMs to predict structural connectivity, it is not entirely known how robust these results are. However, we can compare the correlations to the range of correlations that previous studies reported when doing the other way with the BNM: predicting function from structure. These correlations were in the range of 0.3 to 0.7 (Senden et al., 2017; Cabral et al., 2011). So, while not a perfect comparison, it can be seen that the solutions generated here are all in the range of previously published generative models.

However, while these results are very good, we can also discuss the reasons the models did not converge on a better, or even perfect, solution. It is possible that the solution spaces could have had multiple local minima, and the models were not necessarily reaching the global minimum. In other words, as the network ‘walks down’ the solution space of the loss function using gradient descent, it could get ‘stuck’ in a minimum and thus reach a solution that is not the optimal one. This seems unlikely, however, because each run of these models had a different (random) initialization of the matrix, which all converged to a single solution. Also, results were not significantly different even after increasing the learning rate. However, this problem is still possible, since the random initializations, while having different values, were all similar in that they were on the same uniform distribution between 0 and 1 and did not contain any discernable structure or organization. Perhaps, the global solution could only be reached with an initialization with some non-uniform structure. Some ideas to vary the initialization would be to start with either a sparsified random matrix or a single structural connectome with noise added in the non-sparse areas (i.e. the intra-hemispheric areas). That way, the model would start with something with similar structure to the actual brain SC but much noisier in the areas where a lot of the structure is, which could then be tuned by the model to resemble the actual SC. This implementation is closer to our initial idea and should be done in a future study, which may produce better results.

A possible, but unrelated, reason for why the models did not converge perfectly is due to issues with the rs-fMRI data used in this experiment. The functional connectivity image produced from these data seems to have more noise than functional connectivity produced from other data sources, such as the Human Connectome Project (Kashyap et al., 2018), leading to a connectome that is not as ‘sharp’, containing less discernable structure. This could be due to a longer sampling time in this dataset or due to some other factors in the data collection procedure for the MyConnectome data used in this experiment. This is a qualitative observation, however, and the exact extent of the extra noise is not known, however.

Limitations

A few possible limitations of this experiment have already been mentioned, such as the initialization of the model, the possibility of overfitting, and an rs-fMRI dataset that may be of lower quality than that used in previous studies by this lab.

All of the regularizations in these models used the Frobenius norm because it was the default used by NumPy, the Python package used in this experiment for such matrix operations. However, since our weight matrices are sparse, the L1-norm, defined as the absolute column sum of the matrix, might have been a better choice for a regularization penalty. The L1- norm, compared to other norms, will promote a solution with more large residuals, but it will also have many zeros in the solution. This is precisely the type of matrix we were trying to find, so this norm should be used in future studies instead of the Frobenius norm.

Also, the modeling approach used here makes many simplifying assumptions that do not capture the true complexity of the brain. In the construction of the structural connectome, we assumed that all connections were bidirectional. This is a limitation of using tractography to build the structural network, since tractography cannot distinguish directionality. Moreover, estimates of fiber density for connections between regions that have very sharp angles or between regions that are spatially far apart are far lower than the true connectivity between these regions (Bullmore & Sporns 2009).

Also, the machine learning approach used in this experiment may not be ideal for computing the true optimal solution for this problem. We have already discussed possible problems with distinguishing local from global minima, and the simple convex optimization neural network used here may not have been complex enough to search the solution space proficiently. Other optimization methods are discussed in the next section.

Future Directions

An immediate future experiment could be to run these models with different data to see if the results are replicable. Also, more time could be devoted to analyzing the meaning of these different models in terms of brain dynamics. Of course, it would also be beneficial to use a more complex BNM equation, such as the full Firing Rate model equation with the time delay term implemented, or one of the many other BNM equations that have been used in literature.

Additionally, using other, more complex, convex optimization methods to find an optimal solution to this problem could better find the true solution more reliably, more closely resembling true structural connectivity. The MATLAB function "fmincon" finds the solution to a constrained non-linear optimization problem, of which this brain network model problem is an example. This built-in function is much more complex than the simple method used in this experiment and would be a natural next step, now having some idea of the true solution and knowing the limitations of this method. The SciPy function "optimize" performs a similar function using the Python programming language. Another Python package, Mystic, provides more robust methods for constrained non-linear optimization.

<u>Conclusion</u>

This experiment sought to analyze brain activity as a constrained optimization problem, with the hypothesis that the structural connectivity acted as a constraint on resting-state activity and thus functional connectivity. Using a simple BNM equation, we built three models with regularization penalties of varying complexity and performed convex optimization using gradient descent to search for a weight matrix such that the model most closely mimics real rs-fMRI data. Initializing the matrix randomly produced solutions that had increasing correlations with actual structural connectivity as the complexity of the model increased. The most complex model, which included separating the weight matrix and penalizing tract length, performed exceptionally well, producing a weight matrix with correlation of 0.569 with the ground truth structural connectome in the intra-hemispheric areas.

The major conclusion drawn from this experiment is that such machine learning methods can give a good estimate of brain structure. A possible next step is to try to produce an even better solution that very precisely resembles structural connectivity, which may require other optimization methods that should be explored in future studies. However, this experiment did provide meaningful data which supported the validity of brain network models such as this and gave valuable insights into the relationships between brain structure and function.

Appendix: Python Code

Model 1

```python
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import random

learning_rate = 0.005

initial_weights = np.loadtxt('weight_matrix_19_ordered.csv')
weights_actual = np.loadtxt('weight_matrix_average_15.csv')
real_data = weights_actual.flatten()

# format the data
filename = 'data.txt'
data = np.genfromtxt(filename, delimiter =' ')
data = np.reshape(data, [66, 84, 518])


tr = 1.16
c_prime = np.ones([66,66]) - np.identity(66)

Y = np.diff(data) / tr
X = data[:, :, :517]

#2 placeholders to feed data into
x_placeholder = tf.placeholder(tf.float64, shape=(66, 517))
y_placeholder= tf.placeholder(tf.float64, shape=(66, 517))

# weight matrix: what we are actually changing
# weights = tf.Variable(initial_weights, name='weights', dtype=tf.float64)
weights = tf.Variable(np.random.rand(66,66), name='weights', dtype=tf.float64)


c = np.ones([66,66]) - np.identity(66)
c_const = tf.constant(c)
weights = tf.abs(tf.multiply(weights, c_const))


#computed derivative
computed = -x_placeholder + tf.matmul(weights, x_placeholder)

loss=tf.reduce_mean(tf.square(tf.subtract(y_placeholder, computed)) + 0.009*tf.norm(weights))

optimizer = tf.train.AdamOptimizer(learning_rate).minimize(loss)

with tf.Session() as sess:

    sess.run(tf.global_variables_initializer())
    for step in range(15000):

        trial = random.randrange(0,84)
        session_loss, _, weight = sess.run(
            (loss, optimizer, weights),
            feed_dict={x_placeholder: X[:, trial, :], y_placeholder: Y[:, trial, :]})

        if step % 1000 == 0 :
            flattened_weights = weight.flatten()
            correlation = np.corrcoef(real_data, flattened_weights)
            print("i", step, "loss", session_loss, "correlation", correlation[1, 0])

    # relative error
    print(np.linalg.norm(weight - weights_actual) / np.linalg.norm(weights_actual))

    # correlation
    flattened_weights = weight.flatten()
    initial_weights_flat = initial_weights.flatten()

    correlation = np.corrcoef(real_data, flattened_weights)
    print("correlation:", correlation)
    correlation = np.corrcoef(real_data, initial_weights_flat)
    print("correlation init:", correlation)

    # get arrays of intra-hemispheric areas
    weight_corr1 = weight[0:32, 0:32].flatten()
    weight_corr2 = np.append(weight_corr1, weight[33:65, 33:65].flatten())

    flattened_weights = weight_corr2
    wa1 = weights_actual[0:32, 0:32].flatten()
    wa2 = np.append(wa1, weights_actual[33:65, 33:65].flatten())
    real_data = wa2

    initial_weights_flat = initial_weights.flatten()

    # intrahemispheric relative error
    print(np.linalg.norm(flattened_weights - real_data) / np.linalg.norm(real_data))

    # intrahemispheric correlation
    correlation = np.corrcoef(real_data, flattened_weights)
    print("correlation:", correlation)

    # show weight matrices
    plt.imshow(weight, cmap='hot', interpolation='nearest')
    plt.colorbar()

    plt.show()
```

Model 2:

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import random

learning_rate = 0.005

# load in resting state data
filename = 'data.txt'
data = np.genfromtxt(filename, delimiter=' ')
data = np.reshape(data, [66, 84, 518])

# load in weight matrices
weights_actual = np.loadtxt('weight_matrix_average_15.csv')
real_data = weights_actual.flatten()

initial_weights = np.loadtxt('weight_matrix_19_ordered.csv')
lengths_tract = np.loadtxt('length_matrix_19_ordered.csv')

# calculate actual derivative
tr = 1.16
c_prime = np.ones([66, 66]) - np.identity(66)

Y = np.diff(data) / tr
X = data[:, :, :517]

# set up network
# placeholders to feed data into
x_placeHolder = tf.placeholder(tf.float64, shape=(66, 517))
y_placeHolder = tf.placeholder(tf.float64, shape=(66, 517))

# weight matrices: what we are solving for
# weights= tf.Variable(initial_weights, name='weights', dtype=tf.float64)
weights = tf.Variable(np.random.rand(66, 66), name='weights', dtype=tf.float64)


# set diagonal to 0
c = np.ones([66, 66]) - np.identity(66)
c_const = tf.constant(c)
weights = tf.abs(weights)
weights = (tf.multiply(weights, c_const))

# calculate estimated derivative
y_fit = -x_placeHolder + tf.matmul(weights, x_placeHolder)

# loss function
length_prime = lengths_tract*0.1
loss = tf.reduce_mean(tf.square(tf.subtract(y_fit, y_placeHolder))) + 0.009 *
tf.norm(tf.multiply(weights, length_prime))

optimizer = tf.train.AdamOptimizer(learning_rate).minimize(loss)

# run the model
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for step in range(15000):  # 15k iterations

        trial = random.randrange(0, 84)
        session_loss, _, weight = sess.run(
            (loss, optimizer, weights),  # graph variable we want to compute
            feed_dict={x_placeHolder: X[:, trial, :], y_placeHolder: Y[:, trial, :]})

        if step % 1000 == 0:
            flattened_weights = weight.flatten()
            correlation = np.corrcoef(real_data, flattened_weights)
            print("i", step, "loss", session_loss, "correlation", correlation[1, 0])

    # relative error
    print(np.linalg.norm(weight - weights_actual) / np.linalg.norm(weights_actual))

    # correlation
    flattened_weights = weight.flatten()
    initial_weights_flat = initial_weights.flatten()

    correlation = np.corrcoef(real_data, flattened_weights)
    print("correlation:", correlation)
    correlation = np.corrcoef(real_data, initial_weights_flat)
    print("correlation init:", correlation)

    # get arrays of intra-hemispheric areas
    weight_corr1 = weight[0:32, 0:32].flatten()
    weight_corr2 = np.append(weight_corr1, weight[33:65, 33:65].flatten())

    flattened_weights = weight_corr2
    wa1 = weights_actual[0:32, 0:32].flatten()
    wa2 = np.append(wa1, weights_actual[33:65, 33:65].flatten())
    real_data = wa2

    initial_weights_flat = initial_weights.flatten()

    # intrahemispheric relative error
    print(np.linalg.norm(flattened_weights - real_data) / np.linalg.norm(real_data))

    # intrahemispheric correlation
    correlation = np.corrcoef(real_data, flattened_weights)
    print("correlation:", correlation)

    # show weight matrices
    plt.imshow(weight, cmap='hot', interpolation='nearest')
    plt.colorbar()

    plt.show()
```

Model 3:

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import random

learning_rate = 0.005

# load in resting state data
filename = 'data.txt'
data = np.genfromtxt(filename, delimiter =' ')
data = np.reshape(data, [66, 84, 518])

# load in weight matrices
weights_actual = np.loadtxt('weight_matrix_average_15.csv')
real_data = weights_actual.flatten()

initial_weights = np.loadtxt('weight_matrix_19_ordered.csv')
lengths_tract = np.loadtxt('length_matrix_19_ordered.csv')

# calculate actual derivative
tr = 1.16
c_prime = np.ones([66,66]) - np.identity(66)

Y = np.diff(data) / tr
X = data[:, :, :517]

# set up network
# placeholders to feed data into
x_placeHolder = tf.placeholder(tf.float64, shape=(66, 517))
y_placeHolder = tf.placeholder(tf.float64, shape=(66, 517))

# weight matrices: what we are solving for
# wpos= tf.Variable(initial_weights, name='wpos', dtype=tf.float64)
wpos= tf.Variable(np.random.rand(66,66), name='wpos', dtype=tf.float64)

# wneg = tf.Variable(initial_weights, name='wneg', dtype=tf.float64)
wneg = tf.Variable(np.random.rand(66,66), name='wneg', dtype=tf.float64)

# set diagonal to 0
c = np.ones([66,66]) - np.identity(66)
c_const = tf.constant(c)

wpos  = tf.abs(wpos)
wneg = tf.abs(wneg)
wpos = (tf.multiply(wpos, c_const))
wneg = (tf.multiply(wneg, c_const))

# calculate estimated derivative
y_fit = -x_placeHolder + tf.matmul(wpos, x_placeHolder) - tf.matmul(wneg, x_placeHolder)

length_prime = lengths_tract*.1


# loss function
loss=tf.reduce_mean(tf.square(tf.subtract(y_fit, y_placeHolder))) +
0.005*tf.norm(tf.multiply(wpos, length_prime)) + \
     0.015*tf.norm(tf.multiply(wneg, 1))


optimizer = tf.train.AdamOptimizer(learning_rate).minimize(loss)


# run the model
with tf.Session() as sess:

    sess.run(tf.global_variables_initializer())
    for step in range(15000): #15k iterations
```

```
        trial = random.randrange(0, 84)
        session_loss, _, weight_pos, weight_neg = sess.run(
            (loss, optimizer, wpos, wneg), #graph variable we want to compute
        feed_dict={x_placeHolder: X[:, trial, : ], y_placeHolder: Y[:, trial, :]})

        if step % 1000 == 0:
            weight = weight_pos+weight_neg
            flattened_weights = weight.flatten()
            flattened_pos = weight_pos.flatten()
            flattened_neg = weight_neg.flatten()

            correlation = np.corrcoef(real_data, flattened_weights)
            pos_correlation = np.corrcoef(real_data, flattened_pos)
            neg_correlation = np.corrcoef(real_data, flattened_neg)

            print("i", step, "loss", session_loss, "correlation", correlation[1, 0], " pos
correlation", pos_correlation[1, 0], "neg correlation", neg_correlation[1, 0])

    weight_total = (weight_pos) + (weight_neg)

    # relative error
    print(np.linalg.norm(weight_total - weights_actual) / np.linalg.norm(weights_actual))

    # correlation
    neg_weights = weight_neg.flatten()
    pos_weights = weight_pos.flatten()
    flattened_weights = weight_total.flatten()
    initial_weights_flat = initial_weights.flatten()

    correlation = np.corrcoef(real_data, flattened_weights)
    print("correlation:", correlation)
    correlation = np.corrcoef(real_data, pos_weights)
    print("pos correlation:", correlation)
    correlation = np.corrcoef(real_data, neg_weights)
    print("neg correlation:", correlation)
    correlation = np.corrcoef(real_data, initial_weights_flat)
    #print("correlation init:", correlation)

    # get arrays of intra-hemispheric areas
    weight_corr1 = weight_total[0:32,0:32].flatten()
    weight_corr2 = np.append(weight_corr1, weight_total[33:65,33:65].flatten())

    weight_corr1pos = weight_pos[0:32,0:32].flatten()
    weight_corr2pos = np.append(weight_corr1pos, weight_pos[33:65,33:65].flatten())

    flattened_weights = weight_corr2
    wa1 = weights_actual[0:32,0:32].flatten()
    wa2= np.append(wa1, weights_actual[33:65,33:65].flatten())
    real_data = wa2

    initial_weights_flat = initial_weights.flatten()

    # intrahemispheric relative error
    print(np.linalg.norm(flattened_weights - real_data) / np.linalg.norm(real_data))

    # intrahemispheric correlation
    correlation = np.corrcoef(real_data, flattened_weights)
    print("correlation:", correlation)

    correlation = np.corrcoef(real_data, weight_corr2pos)
    print("correlation:", correlation)

    # show weight matrices
    plt.imshow((weight_pos), cmap='hot', interpolation='nearest')
    plt.colorbar()

    plt.figure()

    plt.imshow((weight_neg), cmap='hot', interpolation='nearest')
    plt.colorbar()
    plt.figure()
    #
    plt.imshow(weight_total, cmap='hot', interpolation='nearest')
    plt.colorbar()

    plt.show()
```

References

Basser, Peter J., et al. “In Vivo Fiber Tractography Using DT-MRI Data.” *Magnetic Resonance in Medicine*, vol. 44, no. 4, 2 Oct. 2000, pp. 625–632., doi:10.1002/1522-2594(200010)44:43.0.co;2-o.

Bihan, D. Le. “Diffusion MRI: What Water Tells Us about the Brain.” EMBO Molecular Medicine, vol. 6, no. 5, 4 Apr. 2014, pp. 569–573., doi:10.1002/emmm.201404055.

Biswal, Bharat, et al. “Functional Connectivity in the Motor Cortex of Resting Human Brain Using Echo-Planar Mri.” *Magnetic Resonance in Medicine*, vol. 34, no. 4, Oct. 1995, pp. 537–541., doi:10.1002/mrm.1910340409.

Breakspear, Michael. “Dynamic Models of Large-Scale Brain Activity.” *Nature Neuroscience*, vol. 20, no. 3, 23 Feb. 2017, pp. 340–352., doi:10.1038/nn.4497.

Bullmore, Ed, and Olaf Sporns. “Complex Brain Networks: Graph Theoretical Analysis of Structural and Functional Systems.” *Nature Reviews Neuroscience*, vol. 10, no. 3, 4 Feb. 2009, pp. 186–198., doi:10.1038/nrn2575.

Cabral, Joana, et al. “Functional Connectivity Dynamically Evolves on Multiple Time-Scales over a Static Structural Connectome: Models and Mechanisms.” *NeuroImage*, vol. 160, 15 Oct. 2017, pp. 84–96., doi:10.1016/j.neuroimage.2017.03.045.

Cabral, Joana, et al. “Role of Local Network Oscillations in Resting-State Functional Connectivity.” *NeuroImage*, vol. 57, no. 1, 1 July 2011, pp. 130–139., doi:10.1016/j.neuroimage.2011.04.010.

Deco, Gustavo, et al. “Emerging Concepts for the Dynamical Organization of Resting-State Activity in the Brain.” *Nature Reviews Neuroscience*, vol. 12, no. 1, 12 Jan. 2011, pp. 43–56., doi:10.1038/nrn2961.

Deco, Gustavo, et al. “The Dynamics of Resting Fluctuations in the Brain: Metastability and Its Dynamical Cortical Core.” *Scientific Reports*, vol. 7, no. 1, 8 June 2017, doi:10.1038/s41598-017-03073-5.

Desikan, Rahul S., et al. “An Automated Labeling System for Subdividing the Human Cerebral Cortex on MRI Scans into Gyral Based Regions of Interest.” *NeuroImage*, vol. 31, no. 3, 1 July 2006, pp. 968–980., doi:10.1016/j.neuroimage.2006.01.021.

Hutchison, R. Matthew, et al. “Dynamic Functional Connectivity: Promise, Issues, and Interpretations.” *NeuroImage*, vol. 80, 2013, pp. 360–378., doi:https://doi.org/10.1016/j.neuroimage.2013.05.079.

Kamagata, Koji et al. “Connectome analysis with diffusion MRI in idiopathic Parkinson's disease: Evaluation using multi-shell, multi-tissue, constrained spherical deconvolution” *NeuroImage Clinical*, vol. 17, 10 Nov. 2017, pp. 518-529., doi:10.1016/j.nicl.2017.11.007

Kashyap, Amrit, and Shella Keilholz. “Dynamic Properties of Simulated Brain Network Models and Empirical Resting-State Data.” *Network Neuroscience*, vol. 3, no. 2, Jan. 2019, pp. 405–426., doi:10.1162/netn_a_00070.

Ogawa, Seiji. “Brain Magnetic Resonance Imaging with Contrast Dependent on Blood Oxygenation.” Proc Natl Acad Sci U S A, vol. 87, no. 24, 1 Dec. 1996, pp. 9868–9872., doi:10.1073/pnas.87.24.9868.

Sanz-Leon, Paula, et al. “Mathematical Framework for Large-Scale Brain Network Modeling in The Virtual Brain.” *NeuroImage*, vol. 111, 1 May 2015, pp. 385–430., doi:10.1016/j.neuroimage.2015.01.002.

Senden, Mario, et al. “Cortical Rich Club Regions Can Organize State-Dependent Functional Network Formation by Engaging in Oscillatory Behavior.” *NeuroImage*, vol. 146, 1 Feb. 2017, pp. 561–574., doi:10.1016/j.neuroimage.2016.10.044.

Shakil, Sadia, et al. “Evaluation of Sliding Window Correlation Performance for Characterizing Dynamic Functional Connectivity and Brain States.” *NeuroImage*, vol. 133, June 2016, pp. 111–128. PMC, doi:10.1016/j.neuroimage.2016.02.074.

Shen, K., et al. “Network Structure Shapes Spontaneous Functional Connectivity Dynamics.” *Journal of Neuroscience*, vol. 35, no. 14, 8 Apr. 2015, pp. 5579–5588., doi:10.1523/jneurosci.4903-14.2015.

Schouten, Tijn M., et al. “Individual Classification of Alzheimer's Disease with Diffusion Magnetic Resonance Imaging.” NeuroImage, vol. 152, 15 May 2017, pp. 476–481., doi:10.1016/j.neuroimage.2017.03.025.

Smith, S. M., et al. “Correspondence of the Brain's Functional Architecture during Activation and Rest.” Proceedings of the National Academy of Sciences, vol. 106, no. 31, 4 Aug. 2009, pp. 13040–13045., doi:10.1073/pnas.0905267106.

Tournier, J. Donald, et al. “MRtrix: Diffusion Tractography in Crossing Fiber Regions.” *International Journal of Imaging Systems and Technology*, vol. 22, no. 1, 14 Feb. 2012, pp. 53–66., doi:10.1002/ima.22005. Accessed 8 May 2018.

Van Essen, David C et al. “The WU-Minn Human Connectome Project: an overview” *NeuroImage* vol. 80, no. 15, 15 Oct. 2013, pp. 62-79., doi:https://doi.org/10.1016/j.neuroimage.2013.05.041

Wolters, Amée F. “Resting-State FMRI in Parkinson's Disease Patients with Cognitive Impairment: A Meta-Analysis.” Parkinsonism & Related Disorders, 17 Dec. 2018, doi:10.1016/j.parkreldis.2018.12.016.