IMPERIAL COLLEGE LONDON

ELECTRICAL AND ELECTRONIC ENGINEERING DEPARTMENT

Interim Report

Project title

STRUCTURE AND DYNAMICS OF LARGE NETWORKS OF INTERACTING NEURONS

Author

ALEJANDRO GILSON CAMPILLO CID: 01112712

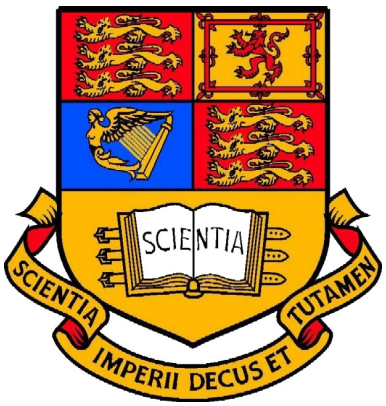
Supervisor

PROF. PIER LUIGI DRAGOTTI

Second Marker

DR. WEI DAI

January 26, 2019



Contents

1 Introduction

The brain is a complex machine, it allows the human being to think, communicate and feel. It does so thanks to the billions of neurons that communicate in a dense network through synapses. However, little is known about how it works. By studying how the neurons structure to store and process information we can understand how the brain as a whole functions. This could have important applications in medicine for curing diseases such as Parkinson [1] and epilepsy [2], and in machine learning for the development of more intelligent neural networks.

In order to infer the network structure of a set of neurons, they are treated as a diffusion network where electrical spikes increase the likelihood of connected neurons to spike and therefore transmit a signal that travels as if it were a disease. By evaluating the time of "infection", the relationship between two neurons can be probabilistically estimated. After computing the relationship between all of the neurons, an estimate of the topology of the network can be obtained.

Previous work on this topic [3, 4] evaluated the feasibility of using a maximum-likelihood estimator algorithm, NetRate [5], for the inference of the structure of biological neural networks. A network was simulated using the Izhikevic neuron model [6] and the Brian simulator [7]. The connections between the neurons were then estimated, compared to the original network and the performance of the algorithm was evaluated.

Recent developments in technology now allow scientists to obtain individual neuron spike information from the brain tissue [8, 9, 10]. This data is very useful and serves as a mean of evaluating the performance of the algorithm with real neurons. Moreover, this information can help in creating simulated networks that resemble more the real biological ones.

2 Project Specification

The aim of this project is to improve on the state of the art research of network inference and the understanding of the underlying structure of the brain. There are many ways in which this can be done such as scalability, increasing the similarity between simulated and real neural networks or changing NetRate so that it is more adapted to the problem in hand. If the simulation were to resemble more the behaviour of a network in a real world scenario it would provide more meaningful information as to how the algorithm would perform with measurements from the brain tissue. Finally, it is of great interest to employ the algorithm for a dataset of real neural network spikes and finding a way of measuring the performance

when no ground truth is given. More information on how this will be achieved will be explained in section 4.

3 Background

3.1 Definition of connectivity

The definition of connectivity between neurons has a history of lack of consensus among the scientific community. Connectivity studies from different researchers may lead to different results depending on how they define it, as they may be looking at different aspects of connectivity. The two main accepted definitions that are used are functional and effective connectivity [11].
Functional connectivity is the temporal correlation between spatially remote neurophysiological events [12]. Studies on this topic began with electroencephalography (EEG) measurements. Some methods to measure functional connectivity include the evaluation of the correlation in the frequency domain between EEG signals at different scalp locations [13], and the cross-correlation of the time series measurements from a pair of electrodes [14]. However, due to the volume conduction of brain tissue, the electrical activity from the scalp cannot infer the individual neuron behaviour below the electrode [11].
Effective connectivity was defined in [12] as the influence that one neural system exerts on another. Effective connectivity can be measured in terms of efficacy and contribution. At a synaptic level it can be expressed as in Eq.1, where x_j is the post-synaptic response to many pre-synaptic inputs x_i and $\mathbf{W}_{ij}$ is the efficacy of the connections between neurons i and j. Contribution is reflected in Eq.2 as the effect of i on j relative to all pre-synaptic inputs. Using this definition, directional effects are taken into account and a richer representation of the network can be attained. Following the approach in [4], this project will focus on the effective connectivity of neurons in a network.

$$x_j = \Sigma \mathbf{W}_{ij} \times x_i \tag{1}$$

$$\frac{\mathbf{W}_{ij}}{\Sigma \mathbf{W}_{ij}} \tag{2}$$

3.2 Izhikevich neuron model

In order to understand how the brain works we must be able to replicate the behaviour of individual neurons applying simple and accurate models. However, as explained in [6],

meeting both criteria can be challenging. The Hodgkin–Huxley model [15] is very accurate as it can emulate the rich firing patterns of many types of neurons. However, it is very computationally expensive and only a few neurons can be computed in real time. The integrate-and-fire model [16] has the opposite problem: it is computationally simple but it is an unrealistic representation of the neuron since it does not capture the firing patterns with sufficient accuracy [6].
In contrast, the Izhikevich neuron model [6] meets both criteria. Tens of thousands of spiking cortical neurons can be simulated in real time by simplifying the Hodgkin-Huxley model into the two dimensional system of differential equations shown below.

$$v' = 0.04v^2 + 5v + 140 - u + I \tag{3}$$

$$u' = a(bv - u) \tag{4}$$

with the auxiliary after-spike reseting

$$\text{if } v \geq 30\text{mV, then } \begin{cases} v & \leftarrow c \\ u & \leftarrow u + d \end{cases} \tag{5}$$

Here, the dimensionless variables v and u represent the membrane potential of the neuron and the membrane recovery, respectively. When a spike reaches its apex (30 mV), both these variables are reset according to Eq. 5. The differentiation is taken with respect to time. Synaptic or injected DC currents are represented by the variable I. Just as with real neurons, the threshold is not fixed and it's based on previous spikes.
On the other hand, a, b, c and d are dimensionless parameters. a determines the speed of the recovery variable u, b defines the sensitivity of the recovery variable u to sub-threshold fluctuations of the membrane potential v. Finally, c and d determine the after-spike reset value of the recovery variables v and u, respectively.
The relevance of this algorithm stems from the fact that, different combinations of the parameters provide the model with a rich variety of firing patterns. When analysing the neocortical neurons in the mammalian brain, a number of different classes of excitatory neurons can be found [17, 18] such as RS (regular spiking), IB (intrinsically bursting) and CH (chattering). From the inhibitory type of neurons, two classes can be found: FS (fast spiking and LTS (low-threshold spiking). Other interesting classes of neurons are the TC (thalamocortical) and the RZ (resonator). A visual representation of these neurons can be observed in figure 1. It is of great importance to understand what types of neurons can be found so that a simulated network can become a closer representation of what can be found on a real brain. In order to simplify the network to be inferred, the only type of neurons simulated in the network were the excitatory regular spiking neurons. This was achieved by setting the

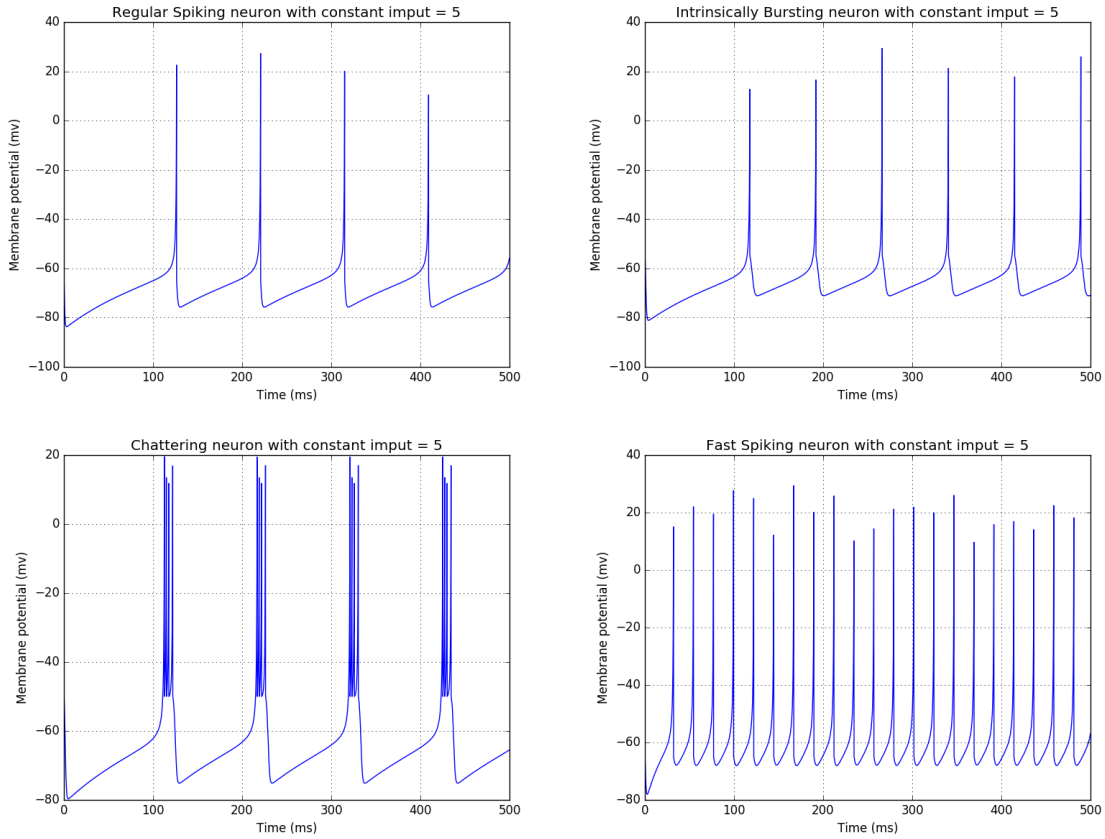


Figure 1: Types of neurons in the mammalian brain. Generated with the Brian Simulator [7] using the Izhikevich neuron model [6]

parameters to $a = 0.02$, $b = 0.2$, $c = -65$ and $d = 8$. This type of neuron is the most common type of excitatory neuron in the brain. There is also a ratio of excitatory and inhibitory neurons of 4 to 1 in the mammalian brain, respectively [6].

In order to make use of the Izhikevich neuron model, Eq.3 is input to the Brian Simulator. This library computes the membrane potential voltage of the interacting neurons in the network and outputs all of their spiking times. This data will then be used to compute the NetRate algorithm.

3.3 Netrate

3.3.1 Diffusion processes

In order to infer the underlying structure of a network, [4] employed the NetRate algorithm developed by Rodriguez [5] by treating the network as a diffusion process.

The study of diffusion networks is based on the observation of the nodes in a system

when they take a certain action: get infected by a virus, share a piece of information, etc. A problem concerning this kind of studies lies on the fact that we can only understand when and where these nodes propagate but not how or why the do so. An example of this is the propagation of a virus in a population. We can tell who and when somebody got infected but not who infected him. For the rest of this section we will refer to the propagation of an infection as the object of study of the network.

To infer the mechanisms behind diffusion processes the time of infection is analysed. A model needs to be created with some assumptions about the structures that generate diffusion processes:

- The network in a diffusion process is fixed, unknown and directed: Connections do not change in time, it is not known what the connections are and connections are not bilateral.
- Infections are binary, they can only be infected or not infected, no partial infections are considered. For real neurons this means that there is either a spike or there is not.
- Infections across the edges of the network occur independently from one another. The probability of node i being infected by node j is not dependent on what the probability of node k infecting node i is.
- The likelihood of a node a infecting node b at time t is modelled by a probability distribution dependent on a, b and t.
- All infections in a network are observed during a recorded time window. This time frame is called horizon [5]. The larger the horizon, the higher the probability of more infections.

NetRate aims to describe how infections occur during a period of time in a fixed network. This is achieved by finding the optimal network and transmission rates that maximize the likelihood of a set of observed cascades to occur. The mathematical definitions that construct this model will be explained in the following section.

3.3.2 Mathematical definitions

The following definitions in this section are necessary for the construction of the model with which we intend to infer the connectivity of the network. First, the data that is going to be analysed will be defined:

- Observations are carried out on a population of N nodes that have created a set of C cascades $\{\mathbf{t}^1, \cdots, \mathbf{t}^{|C|}\}$.

- Each of the cascades $\mathbf{t}^c$ contains the infection times of all the population within a time period T^c.
- Each of the cascades is an N-dimensional vector with recordings of when the nodes were infected in the cascade. If a node was not infected during the time period $[0, T^c]$, a symbol ∞ is assigned. This does not mean that the node never gets infected.

$$\mathbf{t}^c := (t_1^c, \cdots, t_N^c), \quad t_k^c \in [0, T^c] \cup \infty \tag{6}$$

- For simplicity, $T^c = T$
- Node i is parent of node j if $t_i < t_j$ within the cascade.

The pairwise interactions are to be studied in order to obtain the pairwise transmission likelihood between nodes in the network. It will be assumed that infections can occur at different rates along different edges in the network.

- $f(t_i|t_j, \alpha_{j,i})$ is the conditional likelihood of transmission between nodes j and i. It depends on the infection times (t_i, tj) and pairwise transmission rate $\alpha_{j,i}$.
- A node cannot be infected by a healthy node. Node j, infected at t_j, can only infect node i at time t_i if and only if $t_j < t_i$.
- Transmission rate $\alpha_{j,i} \geq 0$.

The cumulative density function is defined as $F(t_i|t_j; \alpha_{j,i})$ and is obtained from the transmission likelihood. If a node j was infected at time t_j, the probability that node i is not infected by node j by time t_i is given by the survival function of the edge $j \to i$:

$$S(t_i|t_j; \alpha_{j,i}) = 1 - F(t_i|t_j; \alpha_{j,i}) \tag{7}$$

The instantaneous infection rate, or hazard function, of the edge $j \to i$ is the ratio of the transmission likelihood over the survival function as shown in Eq.8.

$$H(t_i|t_j; \alpha_{j,i}) = \frac{f(t_i|t_j; \alpha_{j,i})}{S(t_i|t_j; \alpha_{j,i})} \tag{8}$$

With a complete set of definitions, it will now be possible to derive the algorithm behind NetRate as it will be shown in the next section.

3.3.3 Derivation of NetRate

Rodriguez [5] derives NetRate by studying the individual probability of infection of the nodes and then building the whole of the network. The probability of survival of any cascade is the probability that a node is not infected until time T, given that the parents are infected at the beginning of the cascade. For a non-infected node i, the probability that any of the nodes $1 \cdots N$ does not infect node i by time T is given by the product of the survival functions of each of the infected nodes k targeting node i because the different probabilities of infection are considered independent. This is illustrated in Eq.9.

$$\prod_{t_k \leq T} S(T \mid t_k; \alpha_{k,i}) \tag{9}$$

To compute the likelihood of a cascade $\mathbf{t} := (t_1, \cdots, t_N | t_i \leq T)$ we require the the likelihood of the recorded infections $\mathbf{t}^{\leq T} = (t_1, \cdots, t_N | t_i \leq T)$. Again, using independence, the likelihood factorizes as seen in 10. The likelihood of the cascade then becomes the conditional likelihood of the infection time given the rest of the cascade.

$$f(\mathbf{t}^{\leq T}; \mathbf{A}) = \prod_{t_i \leq T} f(t_i \mid t_1, \cdots, t_N \backslash t_i; \mathbf{A}) \tag{10}$$

As in [19], a node gets infected when the first parent infects the node. We now compute the likelihood of a potential parent j of being the first one by using Eq.9.

$$f(t_i \mid t_j; \alpha_{j,i}) \times \prod_{j \neq k, t_k < t_i} S(t_i \mid t_k; \alpha_{k,i}) \tag{11}$$

In this step, we calculate the conditional likelihood of Eq.10 by adding all the likelihoods of the mutually disjoint likelihoods that each potential parent is the first parent:

$$f(t_i \mid t_1, \cdots, t_N \backslash t_i; \mathbf{A}) = \sum_{j: t_j < t_i} f(t_i \mid t_j; \alpha_{j,i}) \times \prod_{j \neq k, t_k < t_i} S(t_i \mid t_k; \alpha_{k,i}) \tag{12}$$

Using Eq.10 and removing the condition $k \neq j$, the likelihood of infections then becomes:

$$f(\mathbf{t}^{\leq T}; \mathbf{A}) = \prod_{t_i \leq T} \prod_{k: t_k < t_i} S(t_i \mid t_k; \alpha_{k,i}) \times \sum_{j: t_j < t_i} \frac{f(t_i \mid t_j; \alpha_{j,i})}{S(t_i \mid t_j; \alpha_{j,i})} \tag{13}$$

However, Eq.13 needs to consider also the nodes that are not infected during the observation window. For this reason we add the multiplicative survival term from Eq.9 and replace the ratios from Eq.13 with hazard functions:

$$f(\mathbf{t}; \mathbf{A}) = \prod_{t_i \leq T} \prod_{t_m > T} S(T \mid t_i; \alpha_{i,m}) \times \prod_{k: t_k < t_i} S(t_i \mid t_k; \alpha_{k,i}) \sum_{j: t_j < t_i} H(t_i \mid t_j; \alpha_{j,i}) \tag{14}$$

The likelihood of a set of independent set of cascades $C = \{t^1, \cdots, t^{|C|}\}$ is the product of the likelihoods of all the individual cascades given by Eq.14:

$$\prod_{\mathbf{t}^c \in C} f(\mathbf{t}^c; \mathbf{A}) \tag{15}$$

The goal of the algorithm is to find the transmission rates $\alpha_{j,i}$ of all the edges in the network such that the likelihood of the set of cascades is maximized.

$$\text{minimize}_{\mathbf{A}} - \sum_{c \in C} \log f(\mathbf{t}; \mathbf{A}) \tag{16a}$$

$$\text{subject to } \alpha_{j,i} \geq 0, i, j = 1, \cdots, N, i \neq j, \tag{16b}$$

.

Here, $\mathbf{A} := \{\alpha_{j,i} \mid i, j = 1, \cdots, n, i \neq j\}$ are the variables and the edges of the network are defined as the pairs of nodes whose transmission rates $\alpha_{i,j} > 0$.

The solution to Eq.16 found in [5] is given by Eq.17a. The survival and hazard functions are concave in the parameter(s) of the transmission likelihoods and, therefore, convexity of Eq.16 follows from linearity. The network inference problem from Eq.16 is thus convex for Power-Law, Rayleigh and Exponential models of the likelihood function.

$$L(\{\mathbf{t}^1 \cdots \mathbf{t}^{|C|}\}; \mathbf{A}) = \sum_c \Psi_1(\mathbf{t}^c; \mathbf{A}) + \Psi_2(\mathbf{t}^c; \mathbf{A}) + \Psi_3(\mathbf{t}^c; \mathbf{A}) \tag{17a}$$

$$\Psi_1(\mathbf{t}^c; \mathbf{A}) = \sum_{i:t_i \leq T} \sum_{t_m > T} \log S(T \mid t_i; \alpha_{i,m}) \tag{17b}$$

$$\Psi_2(\mathbf{t}^c; \mathbf{A}) = \sum_{i:t_i \leq T} \sum_{j:t_j < t_i} \log S(t_i \mid t_j; \alpha_{j,i}) \tag{17c}$$

$$\Psi_3(\mathbf{t}^c; \mathbf{A}) = \sum_{i:t_i \leq T} \log(\sum_{j:t_j < t_i} H(t_i \mid t_j; \alpha_{j,i})) \tag{17d}$$

The terms in Eq.17a depend only on the infection time differences $(t_i - t_j)$ and the transmission rates $\alpha_{j,i}$. Each of the terms adds a property to the solution of NetRate.

- The terms Ψ_1 and Ψ_2 apply a positively weighted norm on $\mathbf{A}$, thus encouraging sparse solutions.
- Ψ_2 penalizes the edges that transmit infections slowly and promotes edges that infect quickly.
- Ψ_1 penalizes edges to uninfected nodes until the time horizon. With a longer observation window the penalties become larger, but so does the probability of nodes becoming infected.

- Ψ_3 makes sure that all infected nodes have a minimum of one parent to avoid $\log 0 = -\infty$. Since the logarithm grows slowly, it slightly encourages infected nodes to have many parents.

3.3.4 Cascade generation

The maximization of the likelihood function requires data in the form of cascades in order to be computed. The time of infection of each of the nodes can be obtained from the network simulation but it must then be formatted into cascades. The rest of this section is based on [3], where cascade generation is explained.

1. At time $t = 0$, a random node is selected to carry the disease.
2. The disease propagates for a T amount of time (horizon) based on the pairwise transmission likelihood $f(t_i \mid t_j; \alpha_{j,i})$ of the edges in the network.
3. At the end of the simulation, a cascade is generated with the information from the times at which the nodes were infected.

As an example, let there be a network of 6 nodes ($N = 6$) and a horizon of $T = 20$. Let us select node 5 at time $t = 0$ to be the starting point of the experiment. The simulation begins and the disease spreads out. It infects node 2 at $t = 3$ and node 6 at $t = 5$. The resulting cascade has the form of Eq.6 would look like this:

$$\mathbf{t}^c = \{\infty, 3, \infty, \infty, 0, 5\} \tag{18}$$

Remember from Eq.6 that the symbol ∞ represents a node that is not infected during the cascade. Nodes 2 and 6 were infected while nodes 1, 3 and 4 remained healthy for the duration of the cascade. Since at time $t = 3$ the only infected node was 3, this node must have infected node 2. However, it becomes inconclusive as to which node infected 6 at time $t = 6$. We could be lead to believe that the uninfected nodes are not connected to any of the other infected nodes. However, cascades are probabilistic models and no one cascade can tell us what the values of $\alpha_{j,i}$ are. We would, therefore, require a large number of cascades in order to infer those values with a high confidence.

3.3.5 Performance metrics

Evaluating the performance of NetRate involves analysing the inferred network $\hat{G}$: which edges have been correctly inferred, which ones have been missed and what weights have been assigned to the inferred edges. These questions are answered in [5] by calculating the accuracy, precision, recall and MAE against the true network G^*:

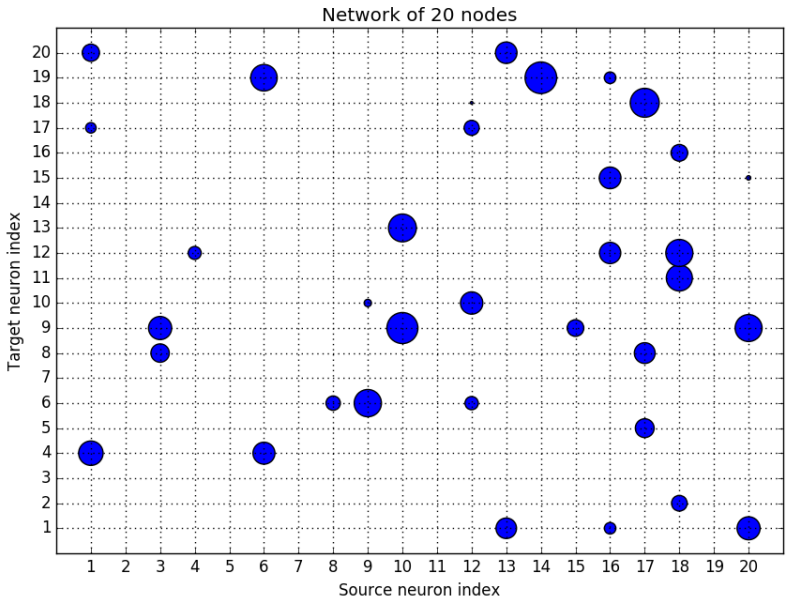


Figure 2: Adjacency matrix of a network of 20 nodes

1. Precision is the proportion of edges in the inferred network that exist in the true network.

2. Recall is the proportion of edges in the true network that exist in the inferred network.

3. Accuracy $= 1 - \frac{\sum_{i,j}|I(\alpha^*_{i,j}) - I(\hat{\alpha}_{i,j}|}{\sum_{i,j} I(\alpha^*_{i,j}) + \sum_{i,j} I(\hat{\alpha}_{i,j})}$, where $I(\alpha) = 1$ if $\alpha > 0$ and $I(\alpha) = 0$, otherwise.

4. MAE $= E[\mid \alpha^* - \hat{\alpha} \mid / \alpha^*]$, where α^* and $\hat{\alpha}$ are the true and estimated transmission rates, respectively.

3.4 Biological Neural Network

3.4.1 Structure of the Network

In order to understand how neural networks are connected, a clear visual representation is required. This is achieved with an adjacency matrix plot. These matrices can be of three different types: binary ($\alpha_{i,j} \in \{0, 1\}$), ternary ($\alpha_{i,j} \in \{-1, 0, 1\}$) or real ($\alpha_{i,j} \in \mathbb{R}$). Due to the characteristics of biological neuron connections, real adjacency matrices are employed. In Fig.2 an adjacency matrix of a network of 20 nodes can be observed. When $\alpha^{BNN}_{j,i} \neq 0$, the target neuron i and the source neuron j are connected and a dot appears on the graph. The source nodes are indexed in the x-axis and the target nodes on the y-axis. The weight of each of the edges is represented by the diameter of the dot.

The superscript BNN in $\alpha_{j,i}^{BNN}$ is employed to differentitate between the weights in biological neural networks and the analogous transmission rates $\alpha_{j,i}$ in diffusion networks [3]. When neurons j and i are connected with a weight $\alpha_{j,i}^{BNN}$, everytime j spikes, it causes the membrane potential v from Eq.3 to increase by $\alpha_{j,i}^{BNN}$. If neuron i crosses the threshold of 30mV, it spikes. This phenomenon is proved in [4] and explained in section 3.4.2. Using the example in figure 2, when node 10 spikes, it causes nodes 13 and 9 to increase by $\alpha_{10,13}^{BNN}$ and $\alpha_{10,9}^{BNN}$, respectively.

The type of networks that were simulated in [4] are generated by Erdös & Reny random graphs, where each possible edge in the network has an independent probability p of being present. The weights assigned to the edges are the output of a uniform distribution in (0,30], the threshold value in Eq.3. It can be observed that the adjacency matrix does not contain weights where $i = j$ because spikes do not increase the membrane potential of the source neuron.

3.4.2 Input stimulus model for cascade generation

The neurons in the brain are susceptible to input stimuli from the rest of the neurons in the network. This is represented in the Izhikevich neuron model with the I component in Eq.3. This term can be employed to model injected current in the form of a DC input or can be normally distributed[1] to represent noise from interactions with neurons that are outside the network being recorded.

The selection of an appropriate input stimulus for the neurons was a challenge encountered by Malhotra [4]. With a Gaussian input I, neurons spike at random times and there is no systematic way of selecting the beginning of a cascade. In a cascade where nodes 1, 3 and 5 spike in chronological order, each of them could have in turn their own cascade (i.e 1, 3 and 5; 3 and 5; and 5). However, this breaks the requirement of independence of cascades seen in section 3.3.1, where no same spike can appear in two cascades. For this reason, a well studied approach was required for the selection of cascades.

The solution Malhotra found to this problem was to provide one neuron at a time with a constant input of 12mV and the rest of the neurons with Gaussian noise. This caused the selected neuron to spike periodically. With an appropriate selection of the DC input, the spiking frequency could be changed, and with a sufficiently long time between spikes, the network could settle into a steady state. Unlike infections in diffusion networks, neurons can spike more than once. For this reason the horizon was arbitrarily limited so as to not allow two spikes from the same cascade to occur in the same cascade and, therefore, obey the law of binary infections imposed by NetRate [4].

[1] In the Izhikevich neuron model [6], the mean and standard deviation are equal to 0 and 5 for excitatory neurons and to 0 and 2 for inhibitory neurons, respectively.

This method provides a systematic way of generating cascades: every time the node with DC input spiked, a new independent cascade was created. In order to obtain cascade information from all the nodes in the network, all nodes are stimulated over the course of an experiment. Otherwise, if only one node was selected, no information would be extracted from the nodes with no direct connection to it [3].

The experimental results obtained by Malhotra show that an optimal amount of spiking information that achieves a high inferring performance is achieved with a stimulation time of 4,000 ms. This is due to the underlying probabilistic nature of NetRate [3]. In other words, more data does not result in a higher performance.

3.4.3 Likelihood function

The ability of NetRate to infer the weights in the adjacency matrix **A** stems from the fact that the shape of $f(t_i \mid t_j; a_{j,i})$ provides a probabilistic description of $\alpha_{j,i}$. The Izhikevich spiking neuron is modelled deterministically while the propagation of infections is probabilistic. For this reason the suitability of NetRate for the biological network inference was proved in [4].

As was explained in section 3.3.4, it is not possible to determine exactly which node caused some other node to become infected. However, this is not true for the Izhikevich neuron spikes. It was shown in [4] that the time it takes from a neuron i becoming unstable to the time it spikes is directly related to $\alpha_{j,i}$. A neuron becomes unstable when it crosses the threshold membrane value of 30mV. This can be caused by another neuron that spikes at that exact time or by random noise. In order to determine the shape that the likelihood function takes for different values of $\alpha_{j,i}$, an histogram of time was employed. It was observed that the likelihood function had an exponential and Rayleigh shape for low and large values of $\alpha_{j,i}$, respectively.Both of these distributions are convex for the solution of the optimization problem in Eq.16. NetRate only allows the use of one model, and because it is more relevant to infer the connections with larger weights, it was decided in [4] to employ the Rayleigh distribution.

3.4.4 Network inference results

The final test to determine the feasibility of the proposed algorithm in [4] was to compare an original simulated network and the resulting inferred network. Due to the underlying probabilistic nature of the network, the ability to infer the connections is different each time the experiment is performed. To provide a better representation of the performance of the algorithm, an average of 10 simulations was made. The results published in [4] can be seen in table 1.

When analyzing the results, it can be observed that the algorithm is good at detecting the edges with large weights from the network since it has a high value for accuracy, precision

	Accuracy	Recall	Precision	MAE
Average performance	0.667	0.633	0.704	0.997
Best performance	0.778	0.7	0.875	0.996
Worst performance	0.596	0.567	0.63	0.994

Table 1: Results for network inference obtained in [4]

and recall. However, the high MAE shows that the algorithm is unable to infer the weights of the edges $\alpha_{j,i}$ correctly. A more extensive explanation for the high MAE can be found in [3].

4 Implementation plan

4.1 Increase the speed of the algorithm

In [4], the size of the studied networks ranged from 10 to 30 nodes with the exception of one experiment with 50 neurons. This was due to the fact that the algorithm is very computationally expensive and it takes a long time for it to provide results. NetRate can be parallelized by nature since it computes an optimization problem for each column in a matrix. However, it makes use of CVX, a package for specifying and solving convex programs [20, 21]. Until this date, CVX cannot be naturally parallelized from within MATLAB.

In [4], a solution to this problem was attempted. Instead of using CVX, a different convex optimization package was employed, namely, CVXPY. This package was written for Python and could be easily parallelizable, so it was a good candidate for replacing CVX. However, solving the optimization problem involved the computation of a logarithmic function and most convex optimization packages are unable of doing so. They make use of successive approximation heuristic by which a polynomial approximation is input to the algorithm consecutively until the result converges. Unfortunately, this method is not used in CVXPY and, therefore, the output of the algorithm has a significantly poorer performance [3].

In this project a novel approach has been implemented by which the algorithm can be parallelized and the size of the network that can be computed is increased. Instead of attempting to parallelize CVX from within MATLAB, several MATLAB instances are run at the same time from the terminal and thus achieving parallelism. Preliminary results show that this method works. However, it seems to have an asymptotic behaviour with respect to the number of processors that the algorithm makes use of. When employing 4 processors, a network of 10, 20, 30 and 40 neurons takes approximately the same, 67%, 50% and 33% the amount of time to be computed than with just one processor.

The aim of the project with this respect will be to measure accurately how much faster

the algorithm is with different sizes of networks. It is also of great interest to formulate an equation that determines the computation time depending on the number of processors, spikes and the size of the network. This would allow us to understand what network size is feasible to compute.

4.2 Implementation of the algorithm on real data

A new dataset has been made available with real neuron spike information from a mouse somatosensory cortex [8, 9, 10]. This data contains several recordings lasting one hour with an average spiking frequency of 2.1Hz. The minimum, maximum and average number of neurons in each of the recordings is 98, 594 and 309, respectively. An ultimate test to the algorithm would be to infer the connectivity of such a network. However, several problems arise:

1. The input stimulus model of the dataset and that of the simulation do not match. The data contains observations from spikes without stimulation (i.e spontaneous activity) and the algorithm is designed for a DC input stimulus. The shape of the likelihood function could be different and it would, therefore, be necessary to estimate it again for the new model.

2. One of the reasons for using a DC input stimulus was to have a systematic way of selecting the beginning of a cascade. Every time the selected stimulated neuron spiked, a new cascade originated [3]. A new method must be devised for cascade generation in the dataset by which NetRate requirements are met and the cascades contain high quality data.

3. Since there is no ground truth for the connectivity of the network in the dataset, there is no obvious way of evaluating the performance of the algorithm. Some sort of data splitting is necessary for determining how good the algorithm is.

In this project, a solution to each of these issues will be attempted. The dataset also contains information about the location of each of the neurons. It could be interested to evaluate if there is any correlation between connectivity and proximity.

4.3 Increase the similarity between simulated and real neural networks

The more the simulated network resembles a biological one, the better it will be possible to determine how accurate the algorithm would perform on real data. This would prove useful if no other method for evaluating the performance is found. There are several ways this could be done:

1. Introduce inhibitory neurons into the network. In the mammalian brain, there is a ratio of 4 to 1 excitatory to inhibitory neurons [6]. The simulated network in [4] only includes excitatory neurons and, therefore, it is not an accurate representation of the biological network.

2. The network can also be made more reallistic by adding different types of excitatory and inhibitory neurons. The only ones considered in [4] were the regular spiking (RS) neurons. However, as discussed in section (3.2), there are many different types.

All these changes to the network would require a new method for infering the weights of the adjancency matrix and this could prove to be challenging. For this reason, the feasibility of this option will only be considered when other, more important milestones, are completed.

4.4 Estimated timeline of the project

The date of submission of the report is in mid-June. This gives the project a time frame of almost 5 months. This is sufficient time for a succesful completion of the project. However, a timeline, like the one seen in table 2, is required in order not to fall behind.

Month	Milestones
February	Evaluate the increase in speed due to parallelism. Create a function that relates the computation time of the algorithm to the number of nodes.
March	See if there are more adequate datasets. If not, create an algorithm for cascade selection under the new input stimulus
April	Simulate the network under the new conditions and assess the performance. Investigate if the shape of the likelihood function needs to be changed.
May	Apply the algorithm to the dataset and draw the inferred network. Find a way of evaluating the performance of the algorithm.
June	Increase the complexity or of the simulated network

Table 2: Timeline of the project from February to June

5 Ethical, Legal and Safety Plan

5.1 Ethical plan

This is a project that aims to increase human understanding of neural networks. No conflictive resources, abusive labour or sensitive information are used in the development of this project. I, Alejandro Gilson Campillo, thereby confirm to the full extent of my knowledge that this project does not incur in any sort of ethical conflict.

5.2 Legal plan

For the development of this project, the programming languages Python and MATLAB are used. The first is open source but the latter is not. However, as a student of Imperial College London I am provided with a free student license for the duration of my degree. Any person that continues with this project must have the required license.

Two software packages are required, namely, CVX [20, 21] and the Brian Simulator [7]. Both of them are open source for the scope of this project. The only requirement is to cite the authors.

5.3 Safety plan

This project is on is entirely based on software, mathematics and biology theory. No physical components are used except for the computer. Therefore, there does not exist any risk of hazard. However, certain issues need to be considered such as posture, hand pain and sight fatigue. I will, do as much as it is possible to reduce these risks.

References

[1] Kim T. E. Olde Dubbelink et al. "Disrupted brain network topology in Parkinson's disease: a longitudinal magnetoencephalography study". In: *Brain* 137.1 (2014), pp. 197–207. ISSN: 0006-8950.

[2] S.C. Ponten, F. Bartolomei, and C.J. Stam. "Small-world networks and epilepsy: Graph theoretical analysis of intracerebrally recorded mesial temporal lobe seizures". In: *Clinical Neurophysiology* 118.4 (2007), pp. 918–927. ISSN: 1388-2457. DOI: https://doi.org/10.1016/j.clinph.2006.12.002.

[3] Pranav Malhotra. *Estimating the Topology of Networks from Distributed observations*. MEng FYP. Imperial College London, 2017.

[4] Roxana Alexandru et al. "Estimating the Topology of Neural Networks from Distributed Observations". In: *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE. 2018, pp. 420–424.

[5] Manuel Gomez Rodriguez, David Balduzzi, and Bernhard Schölkopf. "Uncovering the temporal dynamics of diffusion networks". In: *arXiv preprint arXiv:1105.0697* (2011).

[6] Eugene M Izhikevich. "Simple model of spiking neurons". In: *IEEE Transactions on neural networks* 14.6 (2003), pp. 1569–1572.

[7] Dan Goodman and Romain Brette. "The Brian simulator". In: *Frontiers in Neuroscience* 3 (2009), p. 26. ISSN: 1662-453X. DOI: 10.3389/neuro.01.026.2009.

[8] Shinya Ito et al. "Spontaneous spiking activity of hundreds of neurons in mouse somatosensory cortex slice cultures recorded using a dense 512 electrode array. CRCNS. org". In: *CRCNS. org* (2016).

[9] Shinya Ito et al. "Large-scale, high-resolution multielectrode-array recording depicts functional network differences of cortical and hippocampal cultures". In: *PloS one* 9.8 (2014), e105324.

[10] AM Litke et al. "What does the eye tell the brain?: Development of a system for the large-scale recording of retinal output activity". In: *IEEE Transactions on Nuclear Science* 51.4 (2004), pp. 1434–1440.

[11] Barry Horwitz. "The elusive concept of brain connectivity". In: *NeuroImage* 19.2 (2003), pp. 466–470. ISSN: 1053-8119. DOI: https://doi.org/10.1016/S1053-8119(03)00112-5.

[12] KJ Friston et al. "Functional connectivity: the principal-component analysis of large (PET) data sets". In: *Journal of Cerebral Blood Flow & Metabolism* 13.1 (1993), pp. 5–14.

[13] Gert Pfurtscheller and Colin Andrew. "Event-related changes of band power and coherence: methodology and interpretation". In: *Journal of clinical neurophysiology* 16.6 (1999), p. 512.

[14] Alan S Gevins et al. "Neurocognitive pattern analysis of a visuospatial task: Rapidly-shifting foci of evoked correlations between electrodes". In: *Psychophysiology* 22.1 (1985), pp. 32–43.

[15] Alan L Hodgkin and Andrew F Huxley. "A quantitative description of membrane current and its application to conduction and excitation in nerve". In: *The Journal of physiology* 117.4 (1952), pp. 500–544.

[16] Anthony N Burkitt. "A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input". In: *Biological cybernetics* 95.1 (2006), pp. 1–19.

[17] Barry W Connors and Michael J Gutnick. "Intrinsic firing patterns of diverse neocortical neurons". In: *Trends in neurosciences* 13.3 (1990), pp. 99–104.

[18] Charles M Gray and David A McCormick. "Chattering cells: superficial pyramidal neurons contributing to the generation of synchronous oscillations in the visual cortex". In: *Science* 274.5284 (1996), pp. 109–113.

[19] David Kempe, Jon Kleinberg, and Éva Tardos. "Maximizing the spread of influence through a social network". In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2003, pp. 137–146.

[20] Michael Grant and Stephen Boyd. *CVX: Matlab Software for Disciplined Convex Programming, version 2.1*. `http://cvxr.com/cvx`. Mar. 2014.

[21] Michael Grant and Stephen Boyd. "Graph implementations for nonsmooth convex programs". In: *Recent Advances in Learning and Control*. Ed. by V. Blondel, S. Boyd, and H. Kimura. Lecture Notes in Control and Information Sciences. `http://stanford.edu/~boyd/graph_dcp.html`. Springer-Verlag Limited, 2008, pp. 95–110.