Short-range Impact of Damage on Object Recognition in a Trained Neuronal Network

Thomas Gaudelet

Somerville College

University of Oxford

A thesis submitted for the degree of

MSc Mathematical Modelling and Scientific Computing

2014–2015

Acknowledgements

First, I want to thank my supervisors Mason Porter and Mikko Kivelä for their precious support and advice throughout this thesis. Our weekly meetings helped me greatly in the development of the project.

I want to express my gratitude to Aaditya Rangan from the University of New York for forwarding the code he developed and used in some of his articles.

I also want to thank Daniel Walters and Simon Stringer from the Oxford Centre for Theoretical Neuroscience and Artificial Intelligence (based within the Department of Experimental Psychology) for taking the time to meet me and for the help they provided.

I would also like to express my gratitude to the MSc course director Kathryn Gillow for her support and patience through the year.

Abstract

We investigate the impact of damage on a neuronal network trained to recognise objects. A neuronal network is formed by a set of nodes that represent neurons, and edges that represent the connections between the neurons. To set the stage, we review some existing models that describe the dynamics of individual neurons. We then focus on Integrate-and-Fire (IF) models that we use for this project. We then discuss the numerical method that we use for our model of interacting IF neurons. We then construct our network using a multilayer-network formalism. We consider a simple multilayer network architecture that represents the interactions between the retina and the primal visual cortex (V1) in the brain. We train the system to recognise objects and differentiate between them using the "continuous transformation learning rule", which is based on relative spike times of pre-synaptic and post-synaptic neurons. To highlight the robustness and stability of the trained neuronal network, we simulate damage affecting the connections in the network and measure the performance of the deteriorated systems. A network has good performances if the neurons in the V1 representation differentiate successfully between different objects. In this situation, a subset of the neurons responds strongly to the stimuli corresponding to one object and weakly to the other. Our study provides preliminary insights on the impact of damage on connections between two neuronal subsystems.

Contents

List of Figures

Chapter 1

Introduction

Over the last decades, the interest for networks study has grown substantially. It is especially sensible when contemplating the rapidly increasing literature on the subject across various fields ranging from social sciences to biology [34] (e.g., the study of social networks or the study of how a disease spread). This recent trend stems from a growing interest in the investigation of complex systems, such as brains, societies, and economies. Furthermore, the development of supercomputers and the increased access to large data sets have allowed studies to become more complicated and simulate systems formed of thousands, millions, or even more interacting components. To understand such systems, it is necessary to gain detailed insights on the components of a system, how they interact with each other, and the resulting dynamics that emerge.

The characteristics and dynamics of a system are intimately linked to the interactions of its elements. Connections between the components can come in various form, for instance, chemical or electrical connections between cells and the type of relations in a social environment (e.g., friends, colleagues, or relatives). Furthermore, a system can also sometimes be subdivided into smaller systems that interact with each other. To formally capture the richness of a system and its connectivity, classical graph theory is not always sufficient, leading to the recent and growing popularity of the *multilayer network* framework that offers a formalism to describe complex architectures and tools to study the interactions in a system [4, 26].

Network science is an important tool in neuroscience and, more specifically, for the study of the neuronal systems [29, 40, 45]. Indeed, the brain operates on different spatial and temporal scales, and it is the interactions of the nerve cells on and across such scales that creates the dynamical patterns of the brain. Hence, network representations are particularly fitted for the study of neuronal systems.

Two types of questions can be identified in the study of neuronal systems [40]. First, what are the functions of a neuronal system? Various approaches can help give insights to answer this question. For example, a common practice is to study the deficits arising from damage in part of a system, highlighting the functions that are lost, and thus pointing towards the role of the system [33]. Additionally, knowledge of the relations between different brain regions is often necessary to understand what functions a system performs.

Second, how are the roles fulfilled? Answering this question requires precise knowledge of the interactions of the system components and of the connections with other regions. Moreover, understanding what is the function of a region does not imply the knowledge of how this role is achieved. For instance, knowing which parts of the brain are implicated in object recognition does not entail mechanistic knowledge of the process. Nor does it explain why an object is recognised even if seen from a different angle. To obtain such mechanistic knowledge, it is necessary to understand and model how each part of the systems compute their tasks. It is in trying to answer such questions that network approaches are particularly useful for neuroscience. The research on this subject is still in its infancy, but it is very active.

This relates to the discussion raised in [27] about the "connectomics", which correspond to a structural description of the human brain as a network. As Kopell points out in [27], knowing what is connected is not sufficient to understand how brains compute their tasks. One needs to know how the regions are connected and how communications occur between them. Kopell introduces the notion of "dynome", which correspond to the study of the dynamical structure of the brain and how it relates to cognition.

In this project, we are interested in emulating object recognition in a neuronal network, by using learning rules in association with arbitrary inputs presented to the system. A learning rule intervenes with the connectivity of a network and tunes it in a way that depends on the response of the system to stimuli. We also investigate the short-range impact of damage to a network on the recognition process.

Outline of the study

To get insights on the processes performed by a neuronal system, the first step is to construct a model to represent the behaviour of a single cell. Various frameworks exist in the neuroscience literature, and the choice of the model depends on the aim of the study [2, 12]. If high biophysical accuracy at the neuronal scale is required, then a model that encompasses the biological characteristics of each cell precisely (e.g., Hodgkin–Huxley model, which we discuss in Section 2.2) offers better insights than simpler models. If a study is interested in phenomena that occur on a larger scale (e.g., synchrony), and as such does not require a precise description of the single cells, then simpler and better-understood models can be sufficient (e.g., firing-rate-based models that are discussed in Section 2.2). In Chapter 2, we review several models of neuronal dynamics, and we detail the Integrate-and-Fire (IF) model, that we will use to describe single neuron behaviour in this thesis.

The second step consists of creating an environment to simulate both the single-cells dynamics and the interactions of the cells. The underlying numerical schemes commonly used for IF models are discussed in Chapter 3 and we detail the scheme implemented in the study.

The next step to complete the description of our neuronal system is to specify the connectivity of the network. In Chapter 4, we introduce the *multilayer network* framework and detail the architecture and interactions of our neuronal system.

Chapter 5 is dedicated to the description of how learning can occur through "synaptic plasticity" in a neuronal network. We implement a version of the continuous transformation (CT) learning. To evaluate the impact of the learning on the networks, it is necessary to measure the performance of the network in response to a stimulus. We use analysis of variance (ANOVA), which we detail in Chapter 5.

In Chapter 6, we simulate two types of damage affecting the connections of our network. We detail our numerical experiments we ran in the course of this project and we discuss our results. In Chapter 7, we give conclusions and leads for future work.

Chapter 2

Neuronal dynamics

Before defining the structure that we will use to describe neuronal networks, we need to discuss the dynamics of individual neurons. In this chapter, we start with a description of a single neuron and give a brief review of the existing mathematical models of neurons. For more extensive reviews see [2, 5, 12, 17, 40]. We conclude this chapter by detailing the Integrate-and-Fire model, and we discuss reasons for selecting this model for the rest of the thesis.

2.1 Representation of a single neuron

A neuron is an electrically excitable cell that transmits information through the body by electro-chemical signalling. Neurons are connected to each other and other types of cells via structures called synapses that permit the transfer of information between the cells. The input signals of a neuron can be traced back either to sensory cells or to other neurons, called *pre-synaptic* since they transmit the information to a synapse, as opposed to the *post-synaptic* neurons that receives information from a synapse [40]. A neuron can generally (although some exceptions exist) be decomposed into three components: the cell body (also known as soma), the dendrites, and the axon (see Figure 2.1).

A synapse operates the connexion between the axon of a pre-synaptic neuron and a dendrite of the post-synaptic neuron. Synaptic processing is a key component of the neuronal system, and it is partly electrical and partly chemical. The

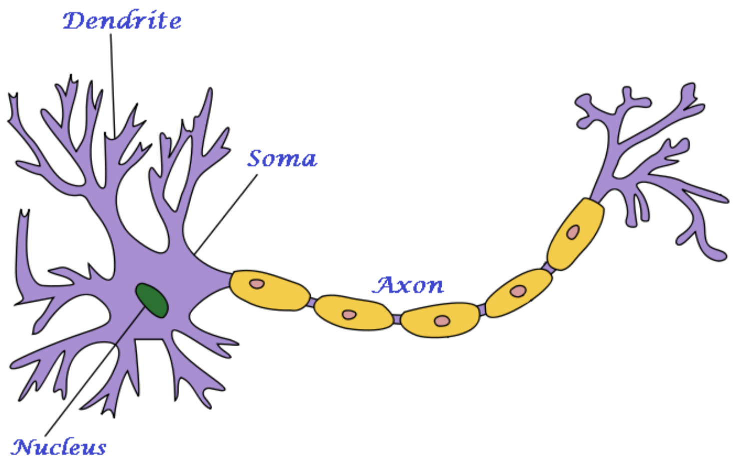


Figure 2.1: Neuron structure. The soma corresponds to the body of the neuron, from which spurts the dendrites and the axon [Creative Commons [22]].

electrical aspect of a neuron is linked to the capacitance and resistance of the neuronal membrane and to the resistivity of the extracellular milieu. The electric field and difference of potential across the membrane around the soma (also known as the *membrane potential*) are controlled by ion channels. These channels can be voltage-gated, their state (open or closed) depends on the level of the membrane potential. Or they can also be chemically-gated, in this case, their state depends on the interaction of chemicals in the extracellular milieu. The existence of ionic channels was first postulated by Hodgkin and Huxley in 1952 [17], and it was confirmed subsequently by experiments led by Katz and Miledi in the 1970s [23–25].

Before moving on to the description of the mathematical models, it is necessary to detail how the ionic fluxes impact the membrane potential [12]. The principal ions involved are the sodium ion (Na^+), the chloride (Cl^-), and the potassium (K^+) ions. *Hyperpolarisation* occurs when either a positive current exits the cell or a negative current enters the cell. In contrast, *depolarisation* takes place when the membrane potential rises with an influx of positively charged ions or an outflux of negatively charged ions. These phenomena are sensitive to the degree of permeability of the membrane to a given ion, which in turn depends on the number of opened ionic channels.

When the membrane potential of a neuron rises sufficiently, it emits a short electrochemical signal, called an action potential, down the axon. The increase in the membrane potential required for the emission of a spike is variable. Once activated, the neuron enters a *refractory period*, during which it cannot fire again and the membrane potential decreases to its resting value at a rate defined by the *membrane time constant*. If the initialisation fails, meaning that the depolarisation of the membrane due to a stimulus is not sufficient to produce an action potential, then the membrane returns to its resting state. Thus, a neuron has a stable resting state and follows an all-or-nothing principle: it fires or resets, depending on the strength of the stimuli. Figure 2.2 represents an idealised time series of the dynamics of a neuron where we assume that the potential threshold for firing is constant.

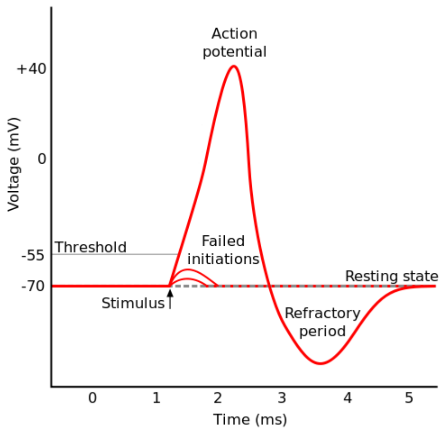


Figure 2.2: Idealised neuron time series. The membrane potential varies in response to stimuli. If at a given time, the stimulus is too weak, then it leads to a failed initialisation, and the membrane voltage returns to the resting value. If, on the contrary, in response to a stimulus, the membrane potential exceeds the threshold, then the neuron fires an action potential down the axon and resets its voltage to the resting value. [Creative Commons [1]]

To summarise, a neuron can be described as a processing cell that adds its inputs (which come from different synapses) and fires an action potential if the sum exceeds a certain threshold. This remark is at the core of numerous mathematical models that have been developed to characterise neuronal networks. To give an order of magnitude, typically a neuron can receive between 5000 and 20000 input connections from other cells [40].

2.2 Neuronal network models

Numerous mathematical models have been proposed to describe the interactions in a network of neurons. Ermentrout and Terman reviewed much of the literature on the subject [12]. Given the omnipresence of oscillations in neuronal systems, models based on coupled oscillators are popular among neuroscientists [2]. The underlying idea is that each node of a network corresponds to a neuron, whose dynamics can be described using some kind of oscillator. The edges of the network represent the connections between the neurons. A model is then described with (1) a set of equations describing the oscillators and (2) a network to represent the interactions.

Before giving a brief review of some existing models for the oscillators, we outline the formalism that we use. Letting $\mathcal{A}$ denote the set of neurons, we index neurons by i. The subset of $\mathcal{A}$ that contains the set of pre-synaptic neurons connected to neuron i is $\mathcal{C}_i$. The strength of the synapses between the neurons is given by the associated weight w_{ij}, the membrane potential of the cell is denoted by V_i, and the firing rate of a neuron is denoted by r_i. In Figure 2.3, we give a rough sketch of this representation.

In the rest of the section, we give a brief overview of some of the existing models describing the *subthreshold* dynamics of a single neuron. We will then give a detailed discussion of the Integrate-and-Fire model that we will use to describe the dynamics of the neuronal networks in this thesis.

2.2.1 Hodgkin–Huxley model

The Nobel Prize winning work of Hodgkin and Huxley on a giant squid axon built the foundation for the modelling of neurons [2, 12, 17]. One of their main discoveries is the existence of voltage-gated conductances. In Figure 2.4, we show a circuit that is equivalent to the neuronal dynamics in the Hodgkin–Huxley (HH) model.

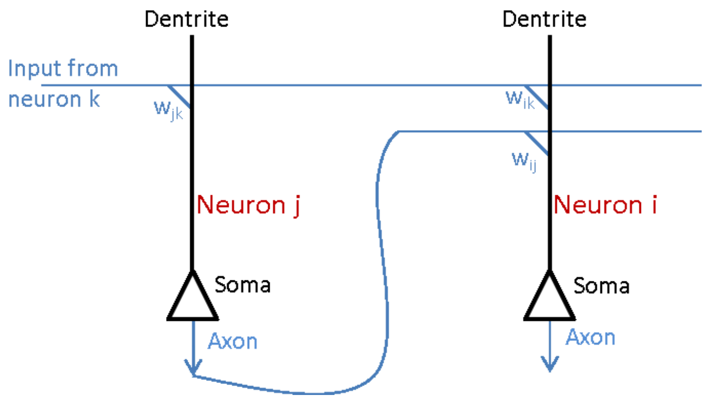


Figure 2.3: Schematic of neurons, denoted i and j, and their coupling. Both i and j receive an input from a neuron k (not represented in the figure), and neuron j is connected to neuron i. In this case, j and k are pre-synaptic neurons to i and k is a pre-synaptic neuron to j.

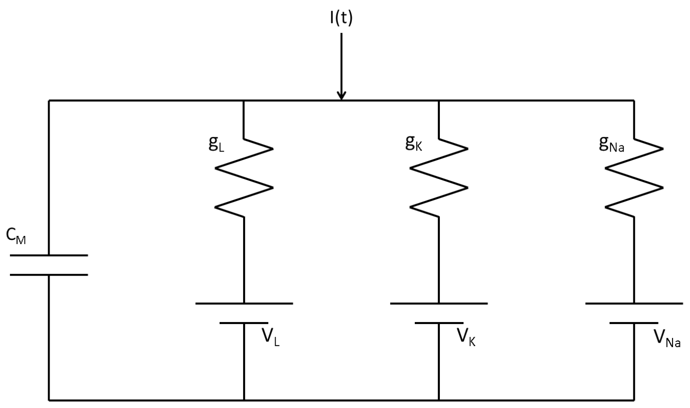


Figure 2.4: Equivalent circuit corresponding to the Hodgkin–Huxley model. The objects labelled by g_X, where $X \in$\{Na,K,L\}, are conductances. The parameter C_M represents the conductance of the soma and E_X, where $X \in$\{Na,K,L\}, are potentials.

In Figure 2.4, the parameter C_M corresponds to the membrane capacitance, $g_{\mathrm{K}} = g_{\mathrm{K}}(t)$ and $g_{\mathrm{Na}} = g_{\mathrm{Na}}(t)$ are the ion conductances (where Na stands for sodium and K stands for potassium), and $g_{\mathrm{L}} = g_{\mathrm{L}}(t)$ is the leak conductance. The potentials V_{L}, V_{K}, and V_{Na} represent the associated voltages of the gated channels.

The governing equation of the model is

$$\begin{aligned} C_M \frac{\mathrm{d}V}{\mathrm{d}t} &= I_{\mathrm{app}} + I_{\mathrm{ion}} = I, \\ I_{\mathrm{ion}} &= -g_{\mathrm{Na}}\left(V - V_{\mathrm{Na}}\right) - g_{\mathrm{K}}\left(V - V_{\mathrm{K}}\right) - g_{\mathrm{L}}\left(V - V_{\mathrm{L}}\right), \end{aligned} \tag{2.1}$$

where I_{ion} is the current generated by the ion fluxes and I_{app} is the applied current, which can comes from a stimulus or the interaction with other neurons.

Hodgkin and Huxley proposed, by fitting experimental data [17], that each K channel has four identical activation gates, the associated conductance depends then on the probability that the four gates are open simultaneously leading to $g_K = \tilde{g}_K n^4$, where $n \in [0, 1]$. In contrast, they proposed that each Na channel depends upon three identical activation (m) and one inactivation gates (h), leading to $g_{\mathrm{Na}} = \tilde{g}_{\mathrm{Na}} m^3 h$, where $(m, h) \in [0, 1]^2$. Here, n, m, and h are gating variables that represent the probability that one of the associated gate is open, they each depend on the membrane potential. The values $\tilde{g}_{\mathrm{K}}$ and $\tilde{g}_{\mathrm{Na}}$ represent the maximal value of the conductances observed when all the gates are open. The gating variables $X \in \{n, m, h\}$ each satisfy a differential equation of the form

$$\frac{\mathrm{d}X}{\mathrm{d}t} = \alpha_X(V)(1 - X) - \beta_X(V)X,$$

where the parameters α_X and β_X were fitted to experimental data by Hodgkin and Huxley resulting in the equations:

$$\begin{aligned} \alpha_n &= \tfrac{0.01(V+55)}{1-\exp(-(V+55)/10)}, & \beta_n &= 0.125\exp(\tfrac{-(V+65)}{80}), \\ \alpha_m &= \tfrac{0.1(V+40)}{1-\exp(-(V+40)/10)}, & \beta_m &= 4\exp\left(\tfrac{-(V+65)}{18}\right), \\ \alpha_h &= 0.07\exp(\tfrac{-(V+65)}{20}), & \beta_h &= \tfrac{1}{1+\exp(-(V+35)/10)}. \end{aligned}$$

Thus, the Hodgkin–Huxley model is a system of four differential equations: one equation describes the membrane potential, and three equations describe the channel-gating variables.

The resulting model exhibits neuron-like behaviour. It has action-potential generation above a certain threshold, a stable resting state for small perturbations, and sustained oscillations for sufficiently high applied current.

2.2.2 Morris–Lecar model

The Morris–Lecar (ML) model [2, 12] is a simplification of the Hodgkin–Huxley model 2.2.1. It is a system of two differential equations. It involves three channels: a leak channel, a potassium channel, and a calcium channel (Ca^{2+}). In the most basic version of the model, the calcium current depends only on the membrane potential (so $g_{\mathrm{Ca}} = \tilde{g}_{\mathrm{Ca}} m(V)$), and the potassium current depends on one activation gate ($g_{\mathrm{K}} = \tilde{g}_{\mathrm{K}} n$). The resulting system is

$$\begin{aligned} C_M \frac{\mathrm{d}V}{\mathrm{d}t} &= I_{\mathrm{app}} - \tilde{g}_{\mathrm{Ca}} m(V) \left(V - V_{\mathrm{Na}}\right) - \tilde{g}_{\mathrm{K}} n \left(V - V_{\mathrm{K}}\right) - g_{\mathrm{L}} \left(V - V_{\mathrm{L}}\right), \\ \frac{\mathrm{d}n}{\mathrm{d}t} &= \frac{n_\infty(V) - n}{\tau_n(V)}, \end{aligned} \tag{2.2}$$

where

$$\begin{aligned} m(V) &= \frac{1}{2}\left(1 + \tanh\left(\frac{V - V_1}{V_2}\right)\right), \\ \tau_n(V) &= \frac{1}{\cosh\left((V - V_3)/2V_4\right)}, \\ n_\infty(V) &= \frac{1}{2}\left(1 + \tanh\left(\frac{V - V_3}{V_4}\right)\right). \end{aligned}$$

The gating variable n approaches the asymptotic value $n_\infty(V)$ with time constant $\tau_n(V)$ and both are voltage-dependent. The parameters V_1, V_2, V_3, and V_4 are chosen to fit the voltage data.

The Morris–Lecar model exhibits characteristic features of neuronal dynamics [2, 12]. It has a stable resting state, spike generation, and a stable limit cycle. The difference with the HH model arises from the simplification of the description of the ionic channels.

2.2.3 Integrate-and-Fire models

In contrast to the HH and ML models, which offer detailed descriptions of single-neuron dynamics, integrate-and-fire (IF) models [2, 5, 12, 40] were developed to study networks of neurons. The neuron dynamics are described by a single variable V associated to the membrane potential (generally, the potential itself) that satisfies an equation of the form

$$\tau\frac{\mathrm{d}V}{\mathrm{d}t} = f(V) + RI(t), \tag{2.3}$$

where $I(t)$ is an external drive and f is a function that depends on the type of model used (see below). The parameters τ and R are associated to the time constant and resistance of the membrane (see Section 2.1).

Associated to Equation (2.3) are two fixed potential values: V_t and V_r, which are for simplicity set to 1 and 0, respectively. This simplification makes the assumption that the fluctuations of the rest and threshold potentials are small enough to be neglected. When $V > V_t$, the neuron is assumed to produce a near-instantaneous action potential at the soma. After spiking, the potential V is reset to V_r. The value of V is clamped to V_r during the refractory period, and its variation is otherwise captured by Equation (2.3).

Various forms of IF models have been studied, and we indicate a few of them.

1. **Simple integrate-and-fire [12, 20]**: The most basic form of the model corresponds to the description of a perfectly integrating cell. Thus, $f \equiv 0$, and Equation (2.3) becomes

$$\tau\frac{\mathrm{d}V}{\mathrm{d}t} = RI(t). \tag{2.4}$$

2. **Leaky integrate-and-fire [12, 20]**: Perhaps the most studied version of the IF model is one that includes a leak term given by $f(V) = -V$. Equation (2.3) then becomes

$$\tau\frac{\mathrm{d}V}{\mathrm{d}t} = -V + RI(t). \tag{2.5}$$

3. **Quadratic integrate-and-fire [11, 12]**: It is defined with the quadratic function $f = V^2$. This yields

$$\tau\frac{\mathrm{d}V}{\mathrm{d}t} = V^2 + RI(t). \tag{2.6}$$

The quadratic IF model offers a description of a certain type of neurons, called θ-neurons [11], that can support low-frequency oscillations [16].

2.2.4 Firing-rate models

In contrast to the previous models, firing-rate-based models do not track the individual spikes of each neuron. Instead, the models track the averaged behaviour of the neurons [12, 40]. They are commonly described by a nonlinear system of the form

$$\begin{aligned} \tau_i \frac{\mathrm{d}V_i}{\mathrm{d}t} &= -V_i + \sum_{j \in \mathcal{C}_i} w_{ij} r_j, \\ r_i &= F(V_i), \end{aligned} \tag{2.7}$$

where the parameter τ_i is a characteristic time constant of the dynamics of the i-th neuron (generally set to be equal to the membrane time constant; see Section 2.1), and the function F expresses the relation between the membrane potential and the firing rate. Different approaches have been used to convert potentials to firing rates [12]. For instance, one can evaluate the firing rate by counting the number of spikes and taking the average over a period of time or by computing the spike's density (i.e., averaging over multiple runs).

If, in the scope of a study, an accurate description of single neuron dynamics is not necessary, there are several practical reasons to opt for a firing-rate-based model [12, 40]. Notably, computational efficiency and analytical tractability. Indeed, when investigating large-scale networks, a conductance-based model that describes precisely the dynamic of each neuron (e.g., the HH and ML models) can become very expensive in computational time and memory use. Moreover, in some experimental setups, the only variable measured is the probability of firing rather than the membrane potential.

2.3 Definition of the Integrate-and-Fire model used

2.3.1 Motivation for studying the Integrate-and-Fire model

In this project, we chose to use an integrate-and-fire model to describe the dynamics of the neurons. Two reasons motivate our choice for using IF neurons. Choosing a model forces a trade-off between biophysical accuracy, computational efficiency, and analytical tractability. We want a model that has both good accuracy at the level of individual neurons and reasonable computational efficiency.

To account for spike-dependent plasticity that will be discussed in Section 5.2, we require a model that tracks individual spikes, which excludes firing-rate-based models. Integrate-and-Fire models offer good compromise between the most accurate models (e.g., the HH model) and the less computational expensive firing-rate-based models.

Furthermore, the simple IF model does not account for the existence of a stable resting state (see Section 2.1), thus we opt for the second type: the leaky IF model (see Equation (2.5)).

2.3.2 Detailed model

The current input $I(t)$ in Equation (2.3) is the sum of the applied current I_{app}, linked to external stimuli, and of the input I_i coming from other neurons. We adopt the following current-based description (as opposed to the conductance-based model used in [13, 36], for instance)

$$\begin{aligned} I_i(t) &= \sum_{Q} \sum_{j \text{ of type } Q} g_{ij}(t) V_Q, \\ g_{ij}(t) &= \sum_{k} w_{ij} \delta(t - t_j^k), \end{aligned} \tag{2.8}$$

where Q represents the different types of neurons (commonly two, corresponding to inhibitory or excitatory neurons), g_{ij} is the conductance of the synapse between the j-th pre-synaptic neuron and the post-synaptic neuron, t_j^k is the firing time of the k-th spike of the j-th pre-synaptic neuron since the last spike time of neuron i, and w_{ij} are the coupling strengths. We define the parameter V_Q as a constant potential associated with the excitatory or inhibitory characteristic of the pre-synaptic neuron. Finally, δ is the Dirac delta function and verifies

$$\delta(t) = \begin{cases} 1, & \text{if } t = 0, \\ 0, & \text{otherwise,} \end{cases} \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(t)\mathrm{d}t = 1.$$

To improve the biophysical accuracy of the model, we add white noise with a mean value of 0 and a standard deviation of μ. Equation (2.3) then becomes

$$\tau \frac{\mathrm{d}V}{\mathrm{d}t} = V_r - V + RI(t) + \mu\sqrt{\tau}\xi(t)\left(V_t - V_r\right), \tag{2.9}$$

where $\xi(t)$ is a Gaussian variable (i.e., $\langle\xi\rangle = 0$ and $\langle\xi(t)\xi(s)\rangle = \delta(t-s)$) representing the Wiener process W, with $\xi(t) = \frac{\mathrm{d}W(t)}{\mathrm{d}t}$. The dimension of ξ is $[\text{time}]^{-1/2}$, where [time] represents the unit of time, so to make the equation consistent, we scale the noise by the square root of the membrane time-constant τ [30].

Chapter 3

Numerical methods for Integrate-and-Fire networks

In this chapter, we briefly review numerical methods that have been used to simulate networks of spiking neurons, hence called *spiking networks.* We then outline the numerical methods that we have implemented and used for this project.

3.1 Literature review

3.1.1 Numerical schemes for single neuron update

Before developing a numerical method to simulate the interactions in a neuronal network, the first step is to choose a numerical scheme to solve for the single neuron dynamics. Different options exist to update the state variables of a neuron when an analytical solution is not available or too complex. The simplest approaches use explicit Euler, implicit Euler, or Runge–Kutta methods. For weakly coupled networks and weak external stimuli (i.e., low values for w_{ij} and I_{app} relatively to the difference $V_t - V_r$), the system of equations is not stiff[1], so standard schemes can be used. An error in computation for one neuron will not have a substantial effect on the rest of the network. However, for other dynamical regimes (e.g.,

[1]A system of equations is said to be "stiff" relative to a numerical scheme when an extremely small time step is required for the stability of the scheme.

strong connections), the system might become stiff and thus require extremely small time steps, which makes the methods computationally expensive.

To tackle this issue, Rangan *et al.* [36][2] proposed a scheme based on the analytical solution of the system of equations (2.3), which they calculated using an integrating factor. One advantage of their method is that it enables the use of a coarse-grained time grid. For a performance review, see [36]. The main advantages are an increased precision and efficiency compared to explicit Euler, implicit Euler, and fourth-order Runge–Kutta methods. The differences with the "classical" methods become more apparent as the network size increases. In contrast, Morrison *et al.* [32] proposed different types of time-constrained methods using exact integration for the subthreshold dynamics of a neuron for IF models.

3.1.2 Numerical methods for spiking networks

Numerical methods for spiking networks are commonly divided into two categories: event-driven methods and time-step-based methods. See [8] for more details. The main method that we use in this project is time-step-based.

3.1.2.1 Event-driven methods

In event-driven methods, sometimes called "asynchronous", one updates the state variables of a neuron only when it either fires or receives an input. Originally, these methods were used for models having an analytical solution [8]. Indeed, updating the state variables of a neuron between the previous update time and the current time is easy when there is a simple formula for the model. For example, in the IF model without incoming spikes, the solution to Equation (2.9) between and initial time t_0 and t (with $V_r = 0$, $V_{\text{thresh}} = 1$) is given by

$$V(t) = V(t_0) \exp\left(\frac{t - t_0}{\tau}\right) + \mu\sqrt{\tau}W(t). \tag{3.1}$$

[2]There are typographical errors in Equation (5) of their article. These played a role in slowing down the progress of our project. Indeed, our initial idea was to implement a code inspired from their paper, and we devoted a lot of time to testing this code and trying to find bugs before we discovered an issue in their paper.

In practice, for such models, highly efficient implementations of event-driven methods exist [7, 50]. However, as the size of a network increases (size corresponding to the number of neurons in the network), one issue arises from the necessity to create a data structure to hold and sort the queue of incoming spikes for a neuron. Indeed, the operations on the queue can become extremely heavy, in terms of memory use, for large systems. Strategies to address these issues have been developed [8, 41]. Some of them make use of a fixed or adaptive time step [9], which in turn allow to extend event-driven methods to models without an analytical solution.

3.1.2.2 Time-step-based methods

The basic idea behind time-step-based methods is to update the neuronal state variables using a fixed time step Δt. These methods are "synchronous", as all neurons are updated at the same time. A compromise has to be made, as high accuracy requires small time steps, while simulation speed requires large time steps. These methods are generally simpler than the event-driven methods to implement, though several issues arise.

The first issue is that an excessively small time step may be required either to achieve sufficiently high accuracy or because the system of equations can become stiff in some dynamic regime (e.g., for the conductance-based IF models used in [13, 36]). Another shortcoming is linked to the spike times. Because the temporal evolution of a neuronal network occurs only at specific times, the occurrence of a spike is assimilated to the nearest temporal grid point [32]. This approach is sufficient for some studies, such as those that examines "high-level dynamics" of a spiking networks [8] i.e., dynamics observable at the level of population rather than single neuron (e.g., synchrony). However for studies that aim at more biophysical accuracy, exact spike times may be required. To tackle this issue, one solution is to increase the temporal resolution. However, this can entail using extremely small time steps and may impact an algorithm's execution time.

To compensate for the above shortcomings, one can use asynchronous time-step-methods that use ideas from the event-driven methods (see Section 3.1.2.1). The time step is adapted in accordance with the activity of the network (see [32, 36]) by letting the system evolve on a given time grid until spikes are detected.

One then determines an approximation of the first spike time through the use of interpolation. If one detects that a neuron's potential has crossed the threshold at a time t_n, then one can assume that the neuron fired in the time interval $[t_{n-1}, t_n]$ and can approximate the spike time by interpolation. Finally, one updates the network at this time point. This approach allows the use of a coarse-grained time grids and speeds up the simulation.

3.2 Numerical method employed in the project

The algorithm that we used is adapted from the one described and used by Zhang *et al.* in their articles [37, 52, 53]. It is based on a simplification of the dynamics of a neuronal network: it ignores delays in neuron–neuron communications. Consequently, when a neuron fires, the impact on the connected neurons is instantaneous. This removes some of the ambiguity of causality in a firing cascade, so it is reasonable to assume that a burst of activity in a network arises from a multi-firing event (i.e., from a cascade of neurons firing). This assumption is reasonable for our purpose as we are not interested in the precise timing between the spikes of a neuron. For a statistical study on the transfer of information and object recognition, the method is appropriate and it gives good insights for how a neuronal network responds to a stimulus. This is one of our primary interests in this thesis. The numerical method is chosen both for its simplicity and adequacy with the objectives of the thesis.

The algorithm provided by [52, 53] uses an adaptive time grid. However, in contrast to the model used by Zhang *et al.* [52, 53], we add Gaussian white noise to the system (see Equation (2.9)), so our spike times can only be approximated. Hence, we use a fixed time step of $\Delta t = 0.01$ ms and the first spike time t_{spk} approximates to the grid point at which the voltage of a neuron has crossed the threshold. Similar to the method proposed by Morrison *et al.* [32], we use the analytical solution to Equation (2.9) to update the state variables of each neuron until a spike is emitted by at least one neuron. Recall that the differential equation associated with the leaky IF model without firing is given:

$$\tau \frac{\mathrm{d}V_i(t)}{\mathrm{d}t} = V_r - V_i(t) + RI_{\text{app}} + \mu\sqrt{\tau}\xi(t)\left(V_t - V_r\right).$$

After the approximation of the first spike time, we freeze the "macroscopic" time t and consider a system that evolves at an arbitrary infinitesimal time scale with time variable $\tilde{t}$. This system is described by the membrane potentials of each neuron $v_i(\tilde{t})$, with $v_i(0) = V_i(t_{\text{spk}}^-)$. Each neuron can only fire once in the infinitesimal time period. Once it has fired, its potential is clamped to the value V_r. That is, it is fixed to this value for the duration of the refractory period. The dynamics at this time scale are described by (see Equation (2.8))

$$\frac{\mathrm{d}v_i}{\mathrm{d}\tilde{t}} = \sum_Q \sum_{j \text{ of type } Q} g_{ij} V_Q, \tag{3.2}$$

$$g_{ij} = w_{ij}\delta(\tilde{t} - \tilde{t}_j), \tag{3.3}$$

where $\tilde{t}_j$ corresponds to the time (if any) in the infinitesimal system at which neuron j fires.

We solve Equations (3.2,3.3) with analytical formulae for each neuron. As an example, consider a neuron at time $\tilde{t}$ with voltage $v_i(\tilde{t})$ and conductance

$$g_i(\tilde{t}) = \sum_{j \text{ of type } Q} g_{ij}(\tilde{t}) V_Q.$$

If this neuron receives no further inputs (and thus does not fire), then at time $\tilde{t}' > \tilde{t}$, we have $g_i(\tilde{t}') = g_i(\tilde{t})$ and $v_i(\tilde{t}') = v_i(\tilde{t}) + \sum_Q g_i(\tilde{t})$. Thus, the voltage membrane v_i has reached an equilibrium. The algorithm evolves the system by processing one spike at a time and updating each neuronal state variable until all of the neurons have either fired or reached an equilibrium. If two neurons cross the threshold at the same time point, then the one closer to the threshold value V_t is assumed to fire first. When the process is finished, we return to the macroscopic time scale and fix the voltages to be $V_i(t_{\text{spk}}^+) = v_i(\infty)$. All firing times are collapsed to t_{spk}, though one can still determine the order of the spikes by looking at the infinitesimal system.

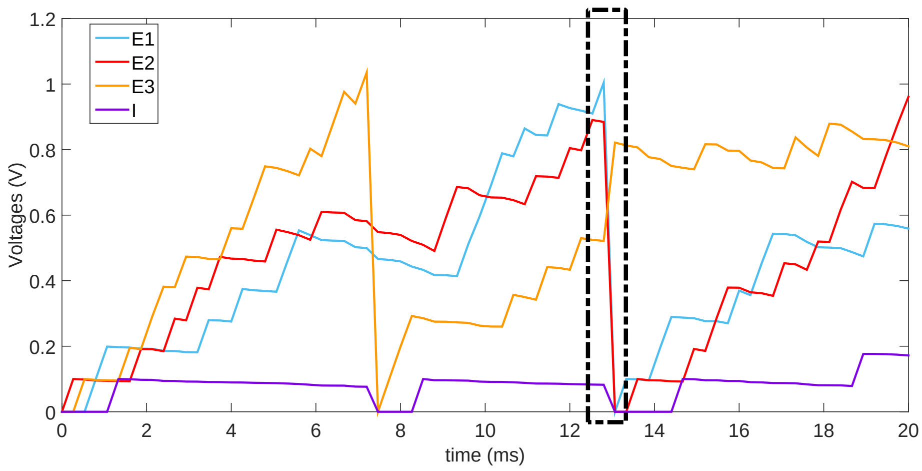


Figure 3.1: Temporal evolution of a simple network formed by three excitatory neurons (E1, E2, and E3) and one inhibitory neuron (I). The temporal order of the events that occur within the black box can be recovered by looking at the infinitesimal time system (see Figure 3.2).

To illustrate the process, Figure 3.1 shows the temporal evolution of an all-to-all connected network formed by three excitatory (E1, E2, and E3) neurons and one inhibitory (I) neuron. In this simulation, we chose the parameters arbitrarily to highlight the algorithm. We fix $V_t = 1$ and $V_r = 0$. The inputs of the network are given by Poisson processes, one for each of the inhibitory and the excitatory population, that have rates of 750 Hz (inhibitory) and 1500 Hz (excitatory). We choose a strong coupling to accentuate the chain of firing: synaptic strengths are fixed to 0.1 for inputs, to 1 for excitatory–inhibitory connections, to 0.34 for excitatory–excitatory couplings, and to 0.37 for inhibitory–excitatory connections. These represent strong couplings, as they are roughly of the same order as the difference $V_t - V_r$.

We observe two spikes during the simulation of the network in Figure 3.1. The first one corresponds to the third excitatory neuron (orange curve), and it leads to a firing of the inhibitory neuron as well, as shown by the purple curve being reset to the resting potential. The second spike occurs in the black rectangle, and we detail the evolution of the associated infinitesimal-time system in Figure 3.2. Here, the excitatory neuron E1 (in light blue) fires first. Its voltage is then clamped to

V_r for the rest of the infinitesimal-time system. This initial spike then causes the membrane potential of E2 (in red) to cross the threshold and fire. This, in turn, leads the inhibitory neuron (in purple) to fire and end the cascade. At the end of the infinitesimal-time system, the potentials of neurons E1, E2, and I are clamped to the rest value V_r, and the last neuron E3 has reached an equilibrium.

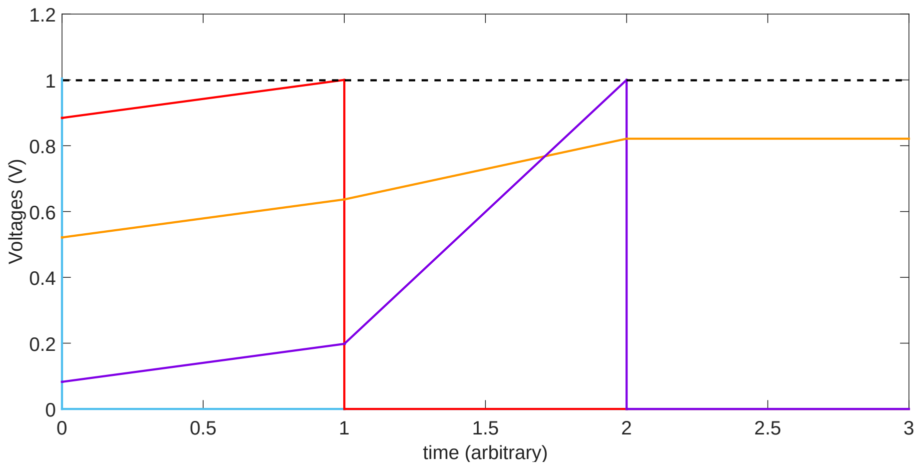


Figure 3.2: Evolution of the infinitesimal-time system of the network formed by three excitatory neurons (E1, E2, and E3) and one inhibitory neuron (I) corresponding to the spikes contained in the black rectangle of Figure 3.1. The colour code is the same that the one defined in Figure 3.1.

Chapter 4

Multilayer structure

In this chapter, we motivate the use of the multilayer structure in networks, and we then define a multilayer network and the associated notations (as introduced in [26]). We conclude by detailing the structure that we use in this project.

4.1 Motivation

Network theory is important for numerous fields of research, such as social and information sciences [34], because it provides frameworks and tools to represent and study interacting systems. In a network, the nodes represent objects, and edges symbolise the interactions between the objects. The structure of connectivity influences the properties of a dynamical system on a network as the structure indicates which components interact with each other. The development of "multilayer network" provides a framework to describe increasingly realistic systems. A real system, whether natural or human-made, generally involves multiple type of entities and/or interactions. For instance, it can involve multiple subsystems or types of connections. Classical graphs (or monolayer network) only consider one type of connection and one type of node, and thus can fail to capture important features of such systems. A multilayer network provides a more comprehensive framework for the description of complex systems. The advantage of the multilayer framework appears in the theoretical description of a system. The formalism that is introduced in Section 4.2 allows a more precise and detailed definition of a system when compared to using graphs, as it becomes easy to distinguish different type of components and interactions. The increasing interest in multilayer networks is striking, as highlighted by the extensive reviews made by Kivelä *et al.* [26] and Boccaletti *et al.* [4], and the numerous papers that have been written since.

The study of dynamical systems on networks seeks to improve the understanding of the relation between dynamical processes that occur in a system and the connectivity of an underlying network of interactions between a system's components. One of the simplest types of phenomena on networks is percolation, and numerous studies have explored how the structure of a multilayer network can affect such processes. Percolation in network theory corresponds to the study of the behaviour of clusters in random graphs. For instance, [39] use percolation processes on a multilayer networks representation of the brain to investigate how connectivity may explain the stability observed in natural systems (the study is based on fMRI data of the brain). In the field of neuroscience, some forms of multilayer networks were introduced several years ago [48, 54, 55], although they did not use the language of multilayer network. Indeed, to develop models and representations in line with experimental results and observation, researchers have been using structure composed of different interconnected populations, where a population is formed of coupled oscillators. For instance, Zhou *et al.* [54, 55] studied synchronisation on a structure inspired by the cortical brain of a cat. Synchronisation in a network of populations of coupled oscillators was also the object of the studies in [3, 43]. Other dynamics were also studied on multilayer networks that were designed to represent part of the visual pathway in the brain of a primate (including orientation selectivity [31], a transform-invariant representation [13], and multi-firing events [38]).

4.2 Notation and general form

In this section, we detail the general form of a multilayer network and introduce the formalism used to describe them. In the literature on the subject, numerous notation has been introduced to characterise their structure. Here, we use the formalism defined by Kivelä *et al.* [26]. The global representation that they proposed encompasses most of the different definitions that exist in the literature.

We start by briefly defining some of the terminology used to describe a *monolayer network* or graph. A graph consists of nodes that are connected by edges. Two nodes connected by an edge are said to be *adjacent*, and the edge is said to be *incident* to each of the two nodes. If two edges are incident to the same node, then they are also said to be incident to each other. A graph is represented by a tuple $G = (V, E)$, where the set V contains the nodes of the graph and the set $E \subseteq V \times V$ gives the edges, where an edge is defined by the pair of nodes that it connects.

For a multilayer network, it is helpful to introduce the notions of *layers* and *elementary layers*. An elementary layer is an element of a set describing one *aspect* of the system studied (e.g., time, type of connections, type of population, etc.). A layer is a combination of elementary layers, one for each aspect. A *multilayer network* is defined by the quadruplet (V_M, E_M, V, L), where V represents the set of nodes in the network and the set L gives the list of sets of elementary layers $(L_1, \ldots, L_d)$, where d is the number of aspects. A layer then belongs to the set defined by $L_1 \times \cdots \times L_d$. The set V_M is defined to clarify the layer(s) in which a given node occurs: $V_M \subseteq V \times L_1 \times \cdots \times L_d$. Note that a node can belong to multiple layers in this formalism. An element of V_M is called a *node-layer-tuple* and an element of V is sometimes called a *physical node*. The set E_M contains the edges of the network. Here, the edges are defined by the pairs of node-layers that it connects: $E_M \subseteq V_M \times V_M$.

Figure 4.1 illustrates the general form of a multilayer network. It contains four nodes, $V = \{1, 2, 3, 4\}$, and two aspects with associated sets L_1 and L_2 of elementary layers, where $L_1 = \{A, B\}$ and $L_2 = \{X, Y\}$. Thus, $V_M = V \times L_1 \times L_2$. Node-layers of this network include $(1, A, X)$ and $(1, B, Y)$, but $(1, A, Y)$ does not occur in this network. The dotted lines represents inter-layer edges (e.g, $((1, A, X), (1, B, X))$), and the solid lines indicate intra-layer edges (e.g, $((4, B, X), (3, B, X))$).

The formalism can be extended further when a system requires a more specific description (see [26]) for instance, by differentiating between inter-layer and

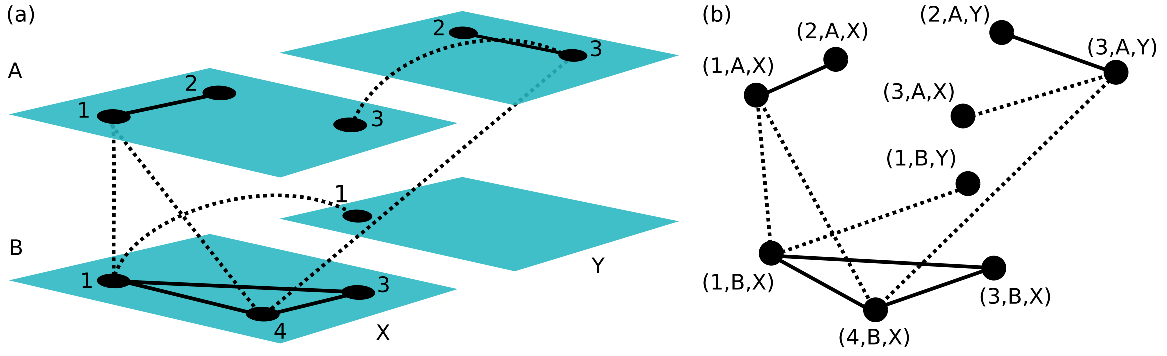


Figure 4.1: (a) Example of a multilayer network. It is composed of four layers ((A,X), (A,Y), (B,X), and (B,Y)) and two types of edges: intra-layer edges (solid lines) and inter-layer edges (dotted lines). (b) The underlying graph $G_M = (V_M, E_M)$ of the multilayer network. [This figure is used with permission from [26].]

intra-layer edges more explicitly. However, for our purpose, the above notions are sufficient to describe the system that we study. However, we do need to define a few more terms for the edges in networks more generally. An edge is *undirected* if information can go both ways. However, considering the structure of a single neuron (see Section 2.1), the edges in a neuronal network are *directed*. They are also *weighted*, as each edge is associated with the strength of the synaptic connection between the pair of neurons. The weight of the directed edge from the neuron j to neuron i corresponds to w_{ij} in Equation (3.3).

4.3 Architecture of the network

The brain is an extremely complex system that contains various numbers of neurons, depending on its size [51]. For example, human brain is estimated to contain between ten billion and a trillion neurons. Commonly, brain systems are decomposed into different components that handle various functions. For example, the role of the hypothalamus is linked to sleep-and-wake cycles, and the medulla is involved in sensory and reflex motor functions such as heart rate. The number of components is linked to the size of an animal (and of its brain). These components can be further decomposed into different intercommunicating regions or layers; see, for instance, the representation of the neocortex in [10] and or the representations of the primary visual cortex (V1) in [15, 31]. Communications between nuclei of different regions or within the same region are extremely intricate; the density, type, and strength of these connections can vary greatly [45].

The multilayer network framework defined in Section 4.2 is an efficient way to represent the wide range of observed, and postulated, structures and interactions. The most common approach is to study a particular region or subsystem of a region [45]. A common object of study is the primary visual cortex (V1) [7, 31, 38]. The V1 region is implicated in pattern recognition [40] and is a subsystem of the visual pathway, which corresponds to the part of the brain that processes visual stimuli. The V1 region is estimated to contain about 140 million neurons [28]. For instance, reference [38] modeled part of the V1. Using the formalism introduced in Section 4.2, the network has 3 aspects. The first one corresponds to "hypercolumns", the second one corresponds to clusters that represent an "orientation preference", and the last one is the type of the neurons (excitatory or inhibitory). The nodes are connected randomly, the probability that an intra-layer edge exist is higher than for an inter-layer edge.

In this project, we also study an architecture inspired by part of the structure of the visual pathway. The basic structure of the network that we use (see Figure 4.2) is the same as the one introduced in [13]. It amounts to simplification of the ventral visual stream, which is important in pattern recognition and memory formation. The network emulates the connection between the retina (input) and group of neurons of the V1 (output)[7]. Visual stimuli are presented to the retina, and the response of the group of neurons is observed.

The architecture of the network is simple: it contains two aspects, which correspond to the sets $L_1 = \{\text{Input}, \text{Output}\}$ and $L_2 = \{\text{Excitatory}, \text{Inhibitory}\}$ of elementary layers. Thus, there are four layers: (Excitatory, Input), (Inhibitory, Input), (Excitatory, Output), and (Inhibitory, Output). To simplify the notation, we denote these layers by E1, I1, E2, and I2, respectively. For clarity, we call the bi-layers defined by {E1,I1} and {E2,I2} (see the dotted boxes in Figure 4.2) the "input cluster" and "output cluster", respectively. In the network, a physical node belongs to exactly one layer. This type of multilayer networks is called "layer disjoint" in [26]. Each arrow in Figure 4.2 represents full connectivity. Thus, all nodes in each inhibitory layer are adjacent to each other (all-to-all intra-layer connections in I1 and I2 layers), and we have all-to-all inter-layer edges in both the input and output clusters. The interactions from the input cluster to the output cluster occur only between the two excitatory layers, and we also assume full connectivity in this case. The inhibitory populations are integrated to introduce "competition" within the excitatory populations. Competition comes from the fact that excitatory neurons that fire first, and cause inhibitory neurons to fire, decrease the potential of the other excitatory neurons. Finally, the weights of the long-range connections evolve throughout the learning phase (see Chapter 5). The competition introduced accentuate the learning; it is an instance of "competitive learning" (see Section 5.2).

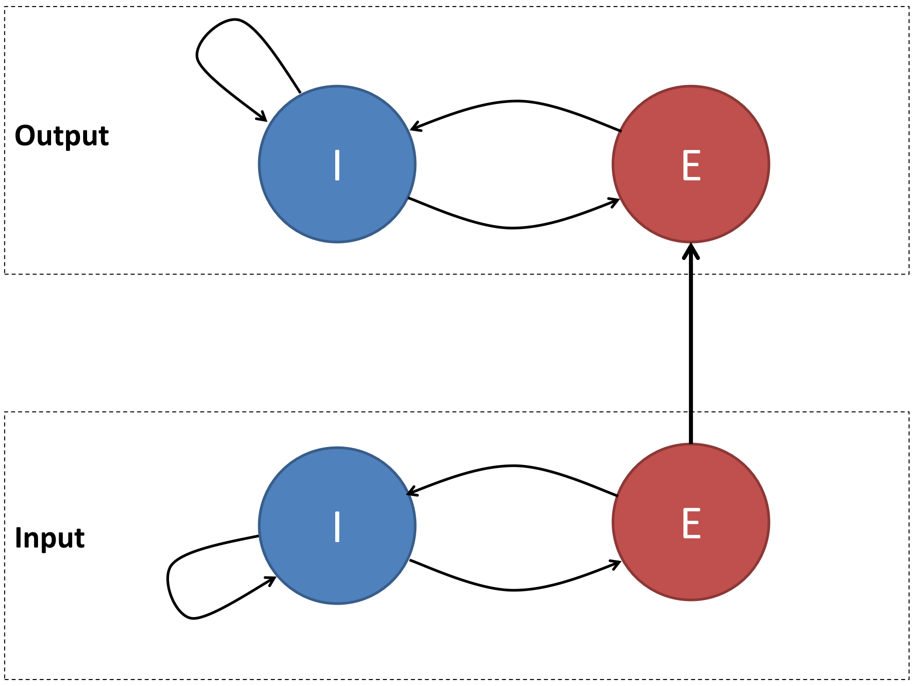


Figure 4.2: Structure of the toy V1 network. The letter E stands for excitatory neurons, and I stands for inhibitory neurons. The network has four layers: (E,Input), (I,Input), (E,Output), and (I,Output).

Algorithm and structure

We now explain how the structure defined in this chapter and the algorithm defined in Chapter 3 fit our objectives. Our main goal is to observe the effect of damage (e.g, lesions) on a network. We use learning to simulate pattern recognition (see Chapter 5) as a way to measure the impact on memory. Network theory is particularly adapted for the study of neuronal interaction and the multilayer formalism allows a clear and concise formal description of how the system behaves. Our study is qualitative, and we thus use a toy model to obtain some preliminary insights on how much information can be lost. The visual pathway is often modelled with four, or more, excitatory layers with feedforward connections [46, 48]. Each of the excitatory layer is associated with an inhibitory layer simulating the mechanism of lateral inhibition observed in real systems. Here, we are interested in how learning affects each layer as well as in the short-range impact of damage on the connectivity. For our purposes, the simpler architecture detailed is sufficient.

A more quantitative approach would require a more realistic network structure and dynamical system. For example, one could want to add delays in transmissions between pairs of nodes (see Chapter 3), and a more complex architecture (e.g., in the number of layers and connections).

Chapter 5

Synaptic plasticity and learning

It is widely accepted that the formation of memory and the learning in a brain is associated with the evolution of the connectivity of a neuronal network and in particular with the modification of the strength of the synapses [14, 40]. The underlying mechanism, called *synaptic plasticity*, refers to the ability of the synapses to strengthen or weaken over time depending on their activity level. In [14], Donald O. Hebb introduced a basic description of such mechanisms that led to the development of the Hebbian theory (see Section 5.2).

We elaborate on the definition of the structure of a synapse (see Section 2.2), before discussing the Hebbian theory. This will help the biophysical interpretation of the learning rule that we implement.

5.1 Synapse structure

In Section 2.2, we introduced the synapse as the junction that operates information transfer between the axon of a pre-synaptic neuron and a dendrite of a post-synaptic neuron. The full details of the synaptic transmission process are not discussed in this thesis, but it is useful to understand the underlying basic steps. In Figure 5.1, we give a the schematic of the structure of a synapse. A synapse consists of three main parts: the axon terminal (or pre-synaptic ending), which contains chemical messengers called neurotransmitters; the synaptic cleft; and the post-synaptic ending (or dendrite terminal), which contains receptors. When an impulse travels down an axon, it triggers the release of neurotransmitters in the synaptic cleft. When enough of the released neurotransmitters link with the receptors on the post-synaptic membrane, it produces an action potential that propagates to the post-synaptic neuron. When there are more unoccupied receptors, more neurotransmitters are required to create an action potential.

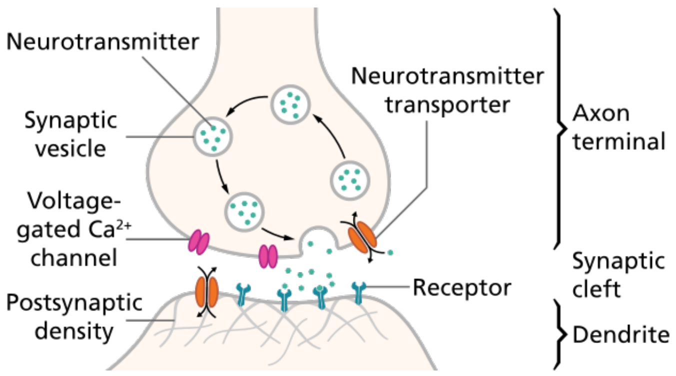


Figure 5.1: Schematic of a synapse between the axon of a neuron and a dendrite of another [Creative Commons [44]].

Essentially, the strength of a synapse is linked to both the number of neurotransmitters contained in the pre-synaptic ending and the number of receptors on the post-synaptic membrane.

5.2 Hebbian theory

To emulate learning in a network, we implement a Hebbian-like method. Hebbian theory was introduced by Donald O. Hebb in 1949 [14]; he formulated the underlying idea as follows:

> Let us assume that the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes that add to its stability. [...] When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

In other words, the wiring between a pre-synaptic and a post-synaptic neuron is strengthened if the pre-synaptic cell contributes to the activity of the post-synaptic, and it is weakened otherwise. The basic concept can be summarised by Carla Schatz's phrase "cells that fire together wire together" [42]. However, one needs to be careful with the phrasing, as it omits the notion of causality present in Hebb's statement. Indeed, the pre-synaptic cell activity needs to "lead" to the firing of the post-synaptic neuron, so it needs to fire first. This is an important point in the continuous transformation (CT) learning method that we use in this project. Furthermore, the competition introduced by the inhibitory populations impacts the learning in the sense that it favours the firing of neurons that respond faster to the stimulus (i.e., the neurons that have stronger connections to the cells stimulated in the input cluster). Indeed, in parallel to the learning rule, the excitatory neurons that fire lead to a decrease of the membrane potential of the other neurons in the same layer. This implies that the neurons that have not fired are less likely to fire, as their connections are weakened and their potentials decrease.

Continuous Transformation learning

The CT method for spiking networks, such as the ones using IF models, was introduced by Perrinet *et al.* [35] as an adaptation of Hebb's principle.

The model in [35] corresponds to a spike-time-dependant synaptic plasticity. An important feature of the model in [35] is that a spike produced by a neuron propagates both down the axon to the pre-synaptic terminal (feedforward) and back to the dendrites' endings (feedback). We denote the relative concentration of synaptic vesicles containing the neurotransmitters in the axon terminal of the synapses between the neuron j and i (see Figure 5.1) with the variable C_{ij}, and we denote the proportion of unoccupied receptors on the post-synaptic ending's membrane with the variable D_i. These variables are impacted by the feedback and feedforward produced when the neuron fires. This yields the following system of equations:

$$\frac{\mathrm{d}C_{ij}(t)}{\mathrm{d}t} = -\frac{C_{ij}(t)}{\tau_C} + \alpha_C\left(1 - C_{ij}(t)\right)\sum_k \delta\left(t - t_j^k\right), \tag{5.1}$$

$$\frac{\mathrm{d}D_i(t)}{\mathrm{d}t} = -\frac{D_i(t)}{\tau_D} + \alpha_D\left(1 - D_i(t)\right)\sum_k \delta\left(t - t_i^k\right), \tag{5.2}$$

where τ_C and τ_D correspond to the decay time constant of the variables C and D. The parameters α_C and α_D are the associated pulse amplitudes. We assume that these four parameters are the same for all neurons in the network. Note that $(C_{ij}, D_i) \in [0,1]^2$ for $(\alpha_C, \alpha_D) \in [0,1]^2$ and for all $t > 0$. Finally, t_j^k corresponds to the time at which the k-th spike of neuron j occurs, and t_i^k is the time of the k-th spike of neuron i.

Qualitatively, Equation (5.1) implies that when the pre-synaptic neuron j fires, the concentration of neurotransmitters increases according to the parameter α_C in each synapse that is connected to the axon. The concentration then decays to 0 at a rate fixed by the time constant τ_C. Similarly, Equation (5.2) signifies that when the post-synaptic neuron fires, the feedback to the dendrites leads to an increase in the number of unoccupied sites. The synaptic strength $w_{ij}(t)$ evolves according to the system

$$\begin{aligned} w_{ij}(t) &= w_{EE}\lambda_{ij}(t), \\ \frac{\mathrm{d}\lambda_{ij}(t)}{\mathrm{d}t} &= \left(1 - \lambda_{ij}(t)\right) C_{ij}(t) \sum_k \delta\left(t - t_i^k\right) \\ &\quad - \lambda_{ij}(t) D_i(t) \sum_k \delta\left(t - t_j^k\right), \end{aligned} \tag{5.3}$$

where w_{EE} corresponds to the maximum possible value of the synapses' weights, and $\lambda_{ij} \in [0,1]$ represents the relative strength of a synapse.

Biophysically, the system formed by Equations (5.1,5.2,5.3) can be interpreted as follows. When a pre-synaptic cells fires, C_{ij} increases and the weight w_{ij} weakens in a way that depends on the recent activity of the post-synaptic cell. The decrease is larger if the neuron fired recently. Alternatively, when the post-synaptic neuron fires, D_i increases and the synapse connections strengthen with the recent activity of the associated pre-synaptic cells. If a pre-synaptic neuron fired recently, then it is likely that it contributed to the firing of neuron i and thus the synapse weight is increased.

Numerically, we solve Equations (5.1,5.2,5.3) using the same method as before (see Section 3.2). We update the variables C_{ij} and D_i using analytical solutions in the absence of spikes. When a spike is detected, the impact is processed in the infinitesimal-time system that we described in Section 3.2. Note that causality in our system implies that a neuron in the E1 layer fires before a neuron in the E2 layer. Hence, Equation (5.1) and the second sum of Equation (5.3) are solved before the first sum and Equation (5.2) to insure time-spike-dependant plasticity. This process happens simultaneously with simulating neurons' firing.

5.3 Stimuli and training procedure

Continuous transformation learning is based on the spatial continuity of the *transforms* of an object [46]. For instance, consider an object that can be observed at different angles. Each angle gives a different visual stimulus, but each corresponds to the same object. Essentially, an object can be associated to different stimuli that we call transforms.

Recall that the network we use represents part of the visual system in a brain. Seeing an object, produces a visual stimulus that is processed by the neuronal system. In our representation, the stimulus produced is applied on the E1 layer of the input cluster, and we are interested in the response of the neurons in the E2 layer of the output cluster. We assume that stimuli corresponding to different objects do not overlap, meaning that the stimuli are applied on different neurons. However, it is reasonable to assume that stimuli associated to one object are similar to each other. Hence, we consider that transforms of an object should have a certain degree of overlap, which means that some number of the neurons that are stimulated in the transforms are shared between the transforms. The degree of overlap is defined relatively to the number of neurons excited by a stimulus. Reference [13] showed that the degree of overlap plays a role in the learning process. As the amount of overlap becomes larger, the association of transforms corresponding to the same object becomes better.

In this project, we consider two objects with three transforms each (see the sketch given on Figure 5.2). The details of the stimuli (number of neurons stimulated by one stimulus and degree of overlap) are discussed in Chapter 6.

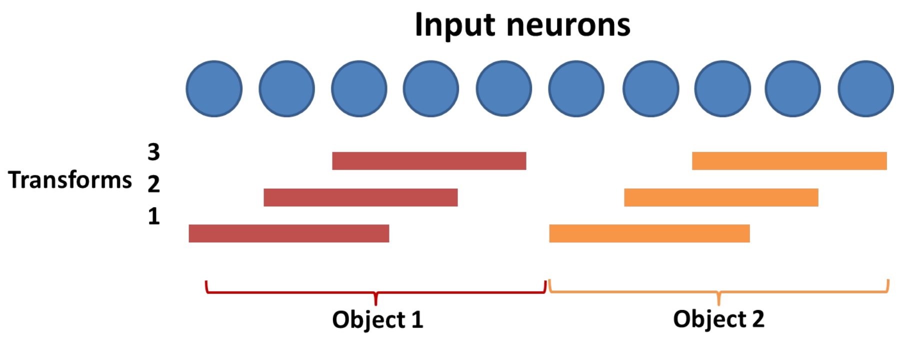


Figure 5.2: Schematic of the objects and their transforms as they are presented to the input excitatory neurons. The neurons excited by each stimulus are directly above the bar that represents it. We represent only 10 neurons to simplify the figure.

A training *sequence* is defined by the successive presentation of the stimuli that correspond to each transform of each object to the input excitatory neurons. Between each transform, the voltages of all neurons, as well as all C_{ij} and D_i, are reset to 0. Reference [13] showed that the decay time constants τ_C and τ_D play an important role for the efficiency of the learning phase. We use the parameters values used in [13, 35]: $\tau_C = 15$ ms and $\tau_D = 25$ ms.

5.4 Performance of the network

We need to be able to measure the performance of the network to evaluate the impact of the learning and the damage to the system. Good performance for a neuron of the network entails *object specificity*, which means that its response to one object, independently of its transform, is stronger than its response to the other object. When the subset of neurons that shows object specificity becomes larger, the performance of the network increases. In this project, to measure performances we consider both a visual method, based on the *raster plots* and scatter plots, and a simple statistical method (the analysis of variance or ANOVA).

A raster plot represents the activity of neurons during a time-frame (see Figure 6.1). The horizontal axis gives the time, and the vertical axis gives the rank of the neurons. Each dot on the graph stands for the corresponding neuron firing. The scatter plots (see Figure 6.4) are defined to show the difference between the responses of the network to two objects and the similarity between the responses of the network to the three transforms of each object. For this, we compute the number of spikes during 100 ms time bins (simulation time) for each stimulus. We then compute the following mean spike counts over 10 runs for each neuron:

1. S_i^{O1} : Mean response to object 1,
2. S_i^{O2} : Mean response to object 2,
3. $S_i^{O1,T1}$: Mean response to the first transform of object 1,
4. $S_i^{O1,T2}$: Mean response to the second transform of object 1,
5. $S_i^{O1,T3}$: Mean response to the third transform of object 1,
6. $S_i^{O2,T1}$: Mean response to the first transform of object 2,
7. $S_i^{O2,T2}$: Mean response to the second transform of object 2,
8. $S_i^{O2,T3}$: Mean response to the third transform of object 2.

A neuron differentiate between objects if its response to one object is high and low to the other (i.e., S_i^{O1} large and S_i^{O2} small, or the opposite). Furthermore, a neuron has an invariant response to the transforms of an object if $S_i^{O,T1}$, $S_i^{O,T2}$, and $S_i^{O,T3}$ are similar for each object. These remarks motivate three scatter plots: one for the responses to both objects, and one for each object and the responses to its transforms.

The ANOVA method was used by Wallis *et al.* [48] to measure the stimulus specificity of neurons. Other available options involve information-theoretic notions; see [6, 29] for details. For instance, [13, 18, 19] used transfer entropy and/or mutual information to evaluate the performance of their networks.

ANOVA is a method that measures differences between sample means in our case, firing rates. Furthermore, ANOVA allows one to differentiate between several independent factors at the same time, and one can thereby evaluate their individual and combined effects on the data sets [21]. We take advantage of this feature of ANOVA. In short, we assume that the firing rates obtained for each neuron while considering different combinations of factors are all equal. To test this hypothesis, we compute a value, denoted F, that corresponds to the ratio between two different measures of the population variance. One measure is based on the hypothesis that the firing rates are equal, and the other measure is only based on the data set. If F exceeds a critical value, then the two measures present sufficient discrepancy, and the hypothesis is rejected. We conclude that there are significant differences among a neuron firing rates for the different combinations of factors. Critical values are drawn from the F-distribution depending both on the desired level of precision (to obtain a confidence interval) and on the degree of freedoms for each factor. In our case, we consider two factors: the object and the transform. Because there are two objects and three transforms, the factor object has 1 degree of freedom and the factor transform has 2 degrees of freedom. We are interested in the separate effect of each factor, so we compute two F-ratios: F_O and F_T. The first is associated to the factor "object" and the second is associated to the factor "transform". A high value of F_O entails a good discrimination between objects. A low value of F_T entails an invariance of responses to the different transforms of an object. The performance of a cell in the network is then measured by the *discrimination ratio* F_O/F_T [48]. The higher the discrimination ratio, the better the performance of the neuron.

In practice, we apply this method for each excitatory cell in the output cluster. We run 20 simulations for each stimulus. A run lasts 100 ms, and we compute the firing rate of each neuron after each run. Once the simulations are done, we use the MATLAB function `anovan` that computes the ANOVA method, and we then plot the discrimination ratio for the E2 layer. We compare the curves that we obtain for the different cases in Chapter 6.

Chapter 6

Simulations and discussion

6.1 Summary of the model

Before detailing our numerical experiments, it is useful to briefly review the equation that we are examining. The sub-threshold dynamics of a neuron i are governed by the equation

$$\begin{aligned}\tau_i \frac{\mathrm{d}V_i(t)}{\mathrm{d}t} &= V_r - V_i(t) + R_i\left(I_i(t) + I_{\mathrm{app}}\right) + \mu\sqrt{\tau_i}\xi(t)\left(V_t - V_r\right), \qquad (6.1)\\ I_i(t) &= \sum_Q \sum_{j \text{ of type } Q} g_{ij}V_Q,\\ g_{ij} &= \sum_k w_{ij}\delta(t - t_j^k).\end{aligned}$$

See Section 2.2.3 for details on the parameters. We assume for simplicity that $\tau_i = \tau$ and $R_i = R$ for all i. The parameter I_{app} corresponds to the stimulus applied to the input excitatory neurons, so $I_{\mathrm{app}} = 0$ for all neurons except for those corresponding to the area of the retina being stimulated, for which we consider $I_{\mathrm{app}} = 2$ nA. For inhibitory–excitatory, excitatory–inhibitory, and inhibitory–inhibitory connections, the weights w_{ij} are constants equal to w_{IE}, w_{EI}, and w_{II}, respectively. See Table 6.1 for the numerical values. The weights of the connections between excitatory populations are governed by the following system (see

Section 5.2 for details):

$$\frac{\mathrm{d}C_{ij}(t)}{\mathrm{d}t} = -\frac{C_{ij}(t)}{\tau_C} + \alpha_C \left(1 - C_{ij}(t)\right) \sum_k \delta\left(t - t_j^k\right), \tag{6.2}$$

$$\frac{\mathrm{d}D_i(t)}{\mathrm{d}t} = -\frac{D_i(t)}{\tau_D} + \alpha_D \left(1 - D_i(t)\right) \sum_k \delta\left(t - t_i^k\right), \tag{6.3}$$

$$w_{ij}(t) = w_{EE}\lambda_{ij}(t), \tag{6.4}$$

$$\begin{aligned}\frac{\mathrm{d}\lambda_{ij}(t)}{\mathrm{d}t} &= \left(1 - \lambda_{ij}(t)\right) C_{ij}(t) \sum_k \delta\left(t - t_i^k\right) \\ &\quad - \lambda_{ij}(t) D_i(t) \sum_k \delta\left(t - t_j^k\right).\end{aligned} \tag{6.5}$$

The architecture that we use contains 72 neurons in the E1 and E2 layers and 18 neurons in the I1 and I2 layers. The choice of a small number of nodes is made to speed the numerical experiments, it is reasonable as we only consider two different objects. We adapted most parameters listed in Table 6.1 from values found in the literature. The other were tuned through successive simulations to obtain results consistent with learning as done in [13]. In particular, it implies that the choice of values for the weights is not unique. Through the tuning phase, we observed that the most important point was how they relate to each other. We then chose values of roughly the same order of magnitude as those found in [13, 35].

The λ_{ij} are all initialised to 0.4, the values evolve during the training phase and are then fixed to their final value for the tests. The other variables in the system are all initialised to 0, and furthermore they are reset to 0 for each test and between stimuli.

In our experiments, we apply each stimulus to 28 neurons of the E1 layer. For one object, we consider that successive transforms have 24 overlapping neurons (i.e., transforms 1 and 2 share 24 neurons, and transforms 2 and 3 share 24 neurons). Furthermore, we consider that transforms of the two objects never overlap. In Figure 6.1 we give the spike raster plot that corresponds to the response of the excitatory input layer to the presentation of each stimulus for 250ms. The first three stimuli correspond to transforms of the first object, and the remaining three are transforms of the second object.

Parameter	**Symbol**	**Value**	**Reference**
Stimulus pulse applied to single neuron	I_{app}	2 nA	
Time-step	Δt	0.01 ms	
Number of neurons in the E1 layer	N_{E1}	72	[13]
Number of neurons in the I1 layer	N_{I1}	18	[13]
Number of neurons in the E2 layer	N_{E2}	72	[13]
Number of neurons in the I2 layer	N_{I2}	18	[13]
Cells membrane resistance	R	0.04 GΩ	[13]
Cells membrane-time constant	τ	20 ms	[13, 47]
Cells firing threshold potential	V_t	1 V	[36]
Cells rest potential	V_r	0 V	[36]
Excitatory neuron constant potential	V_E	14/3 V	[36]
Inhibitory neuron constant potential	V_I	−2/3 V	[36]
Neuron refractory period	τ_{ref}	2 ms	[13, 47]
Synaptic weight for inhibitory–inhibitory connections	w_{II}	40 nS	
Synaptic weight for excitatory–inhibitory connections	w_{EI}	40 nS	
Synaptic weight for inhibitory–excitatory connections	w_{IE}	15 nS	
Synaptic factor for excitatory–excitatory connections	w_{EE}	5 nS	
Synaptic neurotransmitter pulse amplitude	α_C	0.5	[13, 35]
Synaptic receptor pulse amplitude	α_D	0.5	[13, 35]
Post-synaptic time constant	τ_C	15 ms	[13, 35]
Pre-synaptic time constant	τ_D	25 ms	[13, 35]

Table 6.1: Values, symbols, and origin of each of our parameters. The reference column gives the articles from which the values were either taken or adapted. The parameters without reference were tuned, for our purposes, through successive simulations.

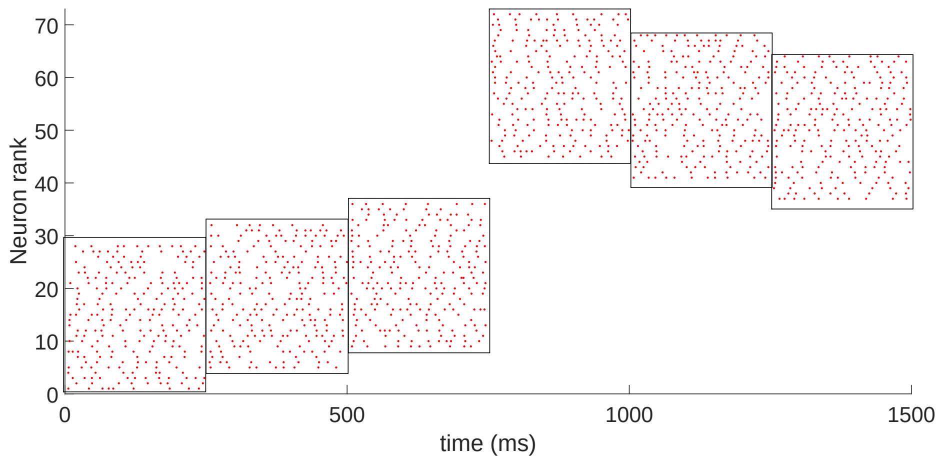


Figure 6.1: Spike raster plot corresponding to the successive presentation of all stimuli. Each stimulus is presented for 250 ms and on 28 neurons corresponding to the boxes; the first three are transforms of the first object, and the latter three correspond to the second object.

6.2 Results and discussion

6.2.1 Training simulation

We train the system over a total of 50 sequences and measure the performance after 2, 20, and 50 sequences. In the following, when we refer to the *trained network*, it corresponds to the state after 50 sequences of training.

Figure 6.2 gives raster plots for the untrained network, and the trained network. Initially, the response of the network to the stimuli is random. However after training, we observe that the activity is more organised and that some cells in the network respond more to the transforms of one object than to the other object. However, we remark that the activity level on the network (i.e., the overall number of neurons spiking), is not significantly different.

In the panel of Figures 6.4, the first striking point is the increase of the variance between the two cases. In Figure 6.4.a, corresponding to the untrained network, we remark that the mean activity of each neuron in response to each object is close

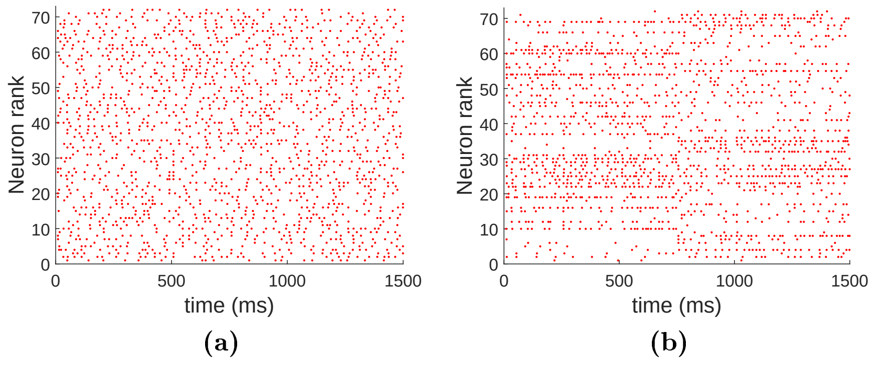


Figure 6.2: (a) Raster plot of the untrained network. (b) Raster plot of the trained network after 50 sequences. The horizontal axis gives the time, the vertical axis gives the rank of the neurons. The red dots represent the spikes of the neurons.

to the overall mean response of the network. However for the trained network, in Figure 6.4.d the variance between the neurons responses increases significantly. Furthermore, the number of neurons that respond strongly to both objects (upper-right quadrant) is lower than in the other regions. This shows that there is an enhanced discrimination between the objects when compared to the untrained network. Ideally, there would be fewer neurons in the bottom-left quadrant. A solution could be to tune down the weights w_{EI} and w_{IE} to reduce the impact of the inhibitory neurons. The remaining figures on the panel underline the invariance of response of the network to the transforms of the same object. In Figures 6.4.b and 6.4.c, we observe that there is no correlation between the transforms of an object in the responses of the untrained network. However, the trained network responses show the a certain degree of invariance, as highlighted by the scatter plots in Figures 6.4.e and 6.4.f.

In Figure 6.5, we give the cumulative plot of the discrimination ratio distribution in the course of the training phase. Recalling that a high discrimination ratio implicates good performance of the corresponding cell, we observe as the training progresses that probability that the cell has a high discrimination ratio increases rapidly. After only two sequences, the network shows good performance; after 20 sequences of training, the network has reached what appears to be its maximal performance. The fact that any further training does not improve performances significantly comes from the noise introduced in the system. The reason the training is efficient so quickly comes from the fact that the network is trained with only two objects with three transforms each. If there were more transforms or more objects the network would require more training sequences to reach a similar level of performance.

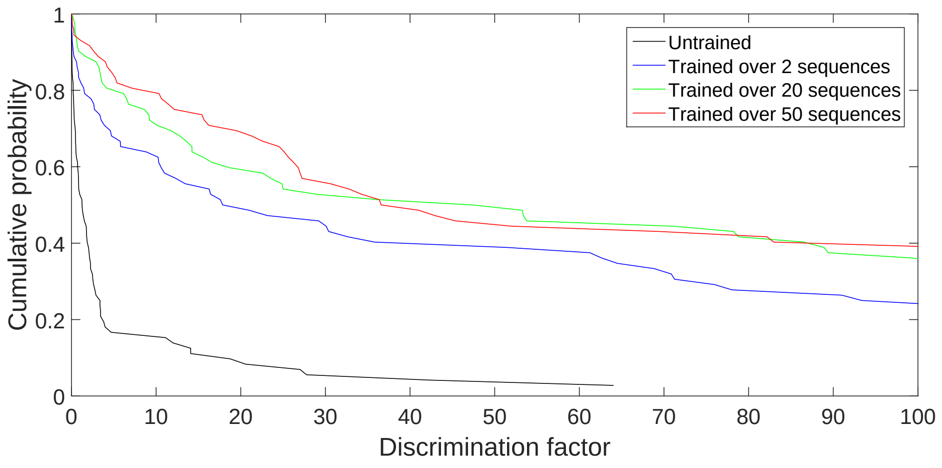


Figure 6.3: Cumulative plot of the discrimination ratio distribution in the output excitatory layer. The vertical axis gives the probability that a neuron has a discrimination ratio above a certain value, which is given on the horizontal axis. The horizontal axis is cut at 100 to improve readability. The black line correspond to the response of the untrained network. The other lines corresponds to the trained network after 2 sequences (blue), after 20 sequences (green) and after 50 sequences (red).

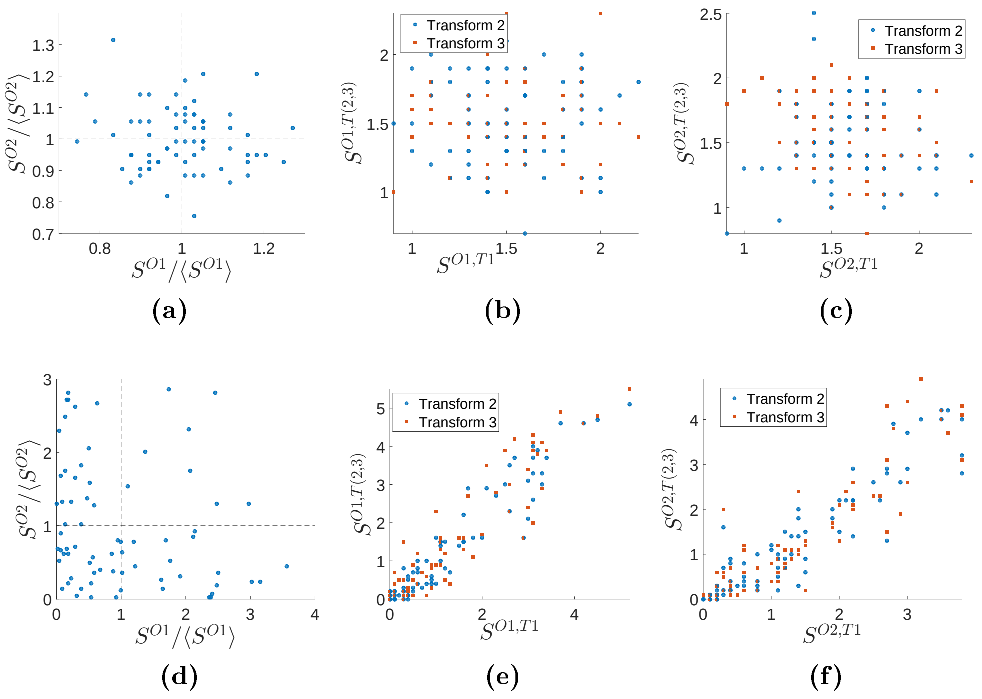


Figure 6.4: These figures correspond to scatter plots. (a), (b), and (c) corresponds to the untrained network, and (d), (e), and (f) are associated with the trained network. (a) and (d) are the scatter plots of the mean spike counts for each cell S_i^{O1} andS_i^{O2} corresponding to responses of the network to object 1 and object 2, respectively. These graphs are normalised with the mean value over all cells for each object $\langle S^{O1}\rangle$ and $\langle S^{O2}\rangle$, which implies that the value 1 on both axis represents the mean response over all cells to each object. (b) and (e) are the scatter plots of the mean spike counts for each cell $S_i^{O1,T1}$, $S_i^{O1,T2}$,and $S_i^{O1,T3}$ corresponding to responses of the network to each transform associated with object 1. $S_i^{O1,T1}$ (blue) and $S_i^{O1,T2}$ (red) are plotted against $S_i^{O1,T3}$. (c) and (f) are the scatter plots of the mean spike counts for each cell $S_i^{O2,T1}$, $S_i^{O2,T2}$,and $S_i^{O2,T3}$ corresponding to responses of the network to each transform associated with object 2. $S_i^{O2,T1}$ (blue) and $S_i^{O2,T2}$ (red) are plotted against $S_i^{O2,T3}$.

6.2.2 Impact of damage

In this section, we test the *robustness* of a trained network to damage. In our case, robustness implies that the performance of a network is maintained, to a certain extent, despite the degradation of the network. In [49], Walters *et al.* highlighted the robustness of firing-rate-based neuronal networks to random removal of nodes in a layer (up to 50%). Here, we are interested in the impact of damage affecting the connections between the E1 and E2 layers. We consider two models of damage on the trained network. The first one corresponds to the destruction of connections that were simulated by removing uniformly at random some percentage of the connections between the excitatory layers. The second type corresponds to the modification of part of the connections, which we simulate by drawing uniformly at random some percentage of the connections and changing their relative weights λ_{ij} to random values drawn from the uniform distribution on the interval $[0, 1]$. Essentially, we are replacing "trained" weights by values drawn uniformly at random.

6.2.2.1 Loss of connections

In Figure 6.5, we show the raster plots after 30% and 70% of the connections were randomly removed from the trained network (the weights of the connections removed are set to 0). We observe two things when we compare Figure 6.2 and Figure 6.5. The first observation is that the activity in the network decreases as the percentage of damage in the network increases (fewer neurons fire in the E2 layer). This observation was expected, as we remove connections between the input and output clusters, without increasing the weights of the remaining edges. Second, we observe that a certain degree of differentiation between the objects is preserved, as shown by the fact that some neuron respond specifically to one object and not to the other.

In the panel of Figures 6.7, the scatter plots give an interesting result: the loss of connections impact the responses between transforms more than the object differentiation. In Figures 6.7.a and 6.7.d, we observe no significant change when compared to the trained network in Figure 6.4.d. This underlines that the neurons in the network conserve, on average, their object specificity. However, the

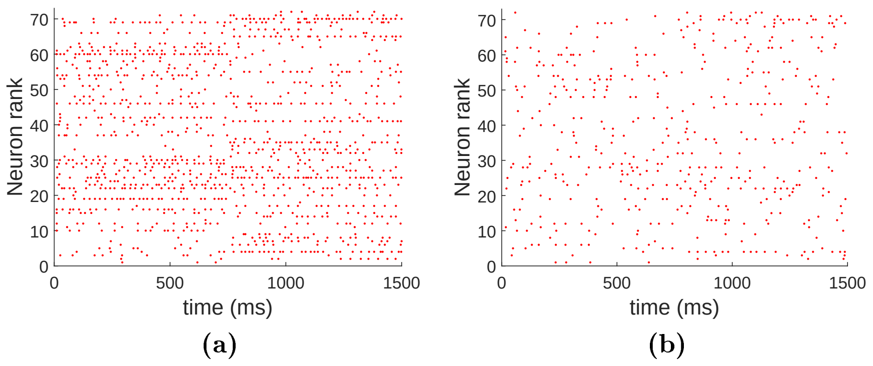


Figure 6.5: (a) Raster plot of the trained network where 30% of the connections were randomly suppressed. (b) Raster plot of the trained network where 70% of the connections were randomly suppressed. The horizontal axis gives the time; the vertical axis gives the rank of the neurons. The red dots represent the spikes of the neurons.

correlation between the responses to each transform of one object decreases as the percentage of damage increases. This is highlighted in Figures 6.7.b, 6.7.c, and particularly in Figures 6.7.e and 6.7.f.

Each curve in Figure 6.9 represents the cumulative plot of discrimination ratio distribution of the neurons in the E2-layer. We remark that the network retains good performance despite the damage. For example, the network with 10% of the connections destroyed has roughly the same distribution that the trained network. With 30% and 50% of the connections removed, the discrimination ratio distribution still shows reasonable performance similar to those obtained after 2 sequences of training on the initial network. However at 70%, we observe that the performance of the network drops considerably.

The study of the destruction of connection between the E1 and E2 layers highlights two results. The first result is the robustness of the network performance to the random removal of edges. On average, the damaged networks appear to differentiate the objects the same way as the trained system. However, this type of damage seems to lead to an increased variance between the responses of the network to the transforms of one object. This result is directly linked to the decrease in activity in the system. It may be interpreted as the network starting to differentiate the objects but also the transforms of an object due to a lack of information reaching the E2 layer (i.e., the network interprets the transforms of an object as independent objects).

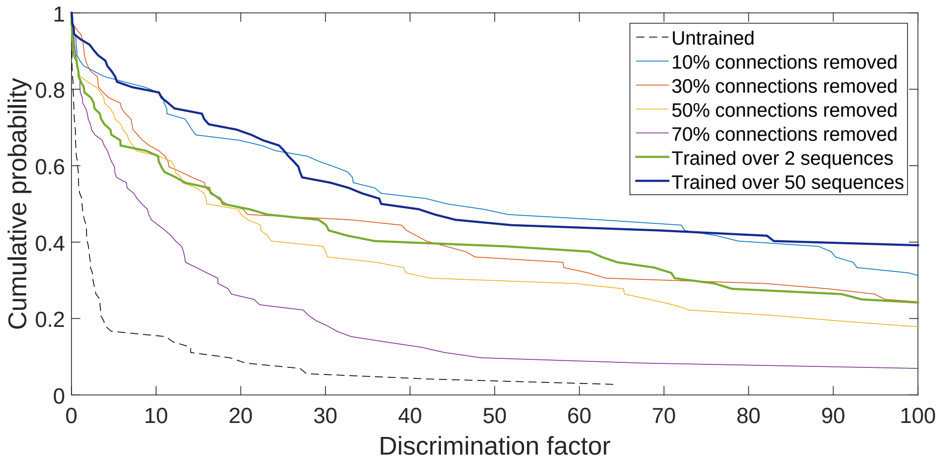
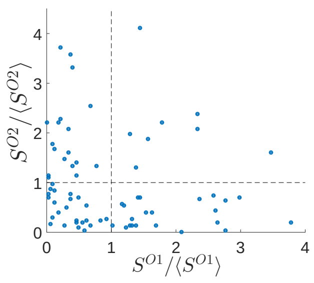
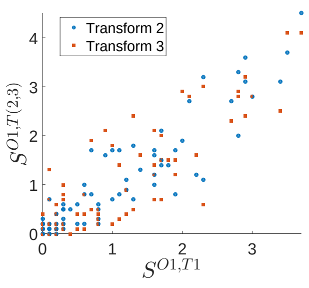


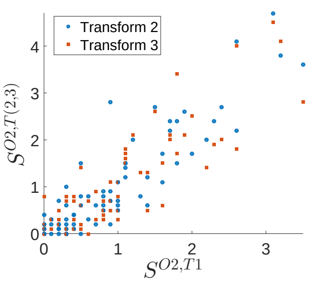
Figure 6.6: Cumulative plot of the discrimination ratio distribution in the output excitatory layer. The vertical axis gives the probability that a neuron has a discrimination ratio above a certain value, which is given on the horizontal axis. The horizontal axis is cut at 100 to improve readability. The dashed line corresponds to the response of the untrained network. The thicker lines correspond to the trained network after 2 sequences (green) and after 50 sequences (blue). The other lines correspond to the trained network after removing randomly 10% (blue), 30% (red), 50% (orange), and 70% (purple) of the connections.



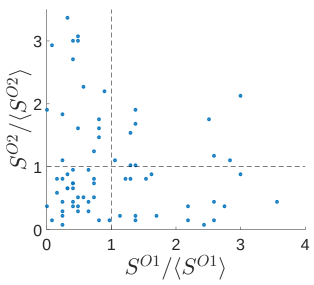
(a)



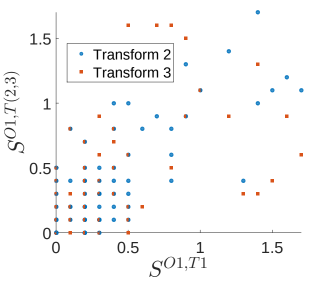
(b)



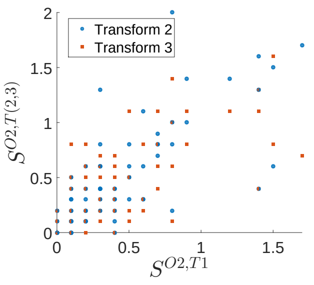
(c)



(d)



(e)



(f)

Figure 6.7: These figures correspond to scatter plots. (a), (b), and (c) corresponds to the trained network with 30% of connections removed, and (d), (e), and (f) are associated with the trained network with 70% of connections down. (a) and (d) are the scatter plots of the mean spike counts for each cell S_i^{O1} andS_i^{O2} corresponding to responses of the network to object 1 and object 2, respectively. These graphs are normalised with the mean value over all cells for each object $\langle S^{O1}\rangle$ and $\langle S^{O2}\rangle$, which implies that the value 1 on both axis represents the mean response over all cells to each object. (b) and (e) are the scatter plots of the mean spike counts for each cell $S_i^{O1,T1}$, $S_i^{O1,T2}$,and $S_i^{O1,T3}$ corresponding to responses of the network to each transform associated with object 1. $S_i^{O1,T1}$ (blue) and $S_i^{O1,T2}$ (red) are plotted against $S_i^{O1,T3}$. (c) and (f) are the scatter plots of the mean spike counts for each cell $S_i^{O2,T1}$, $S_i^{O2,T2}$,and $S_i^{O2,T3}$ corresponding to responses of the network to each transform associated with object 2. $S_i^{O2,T1}$ (blue) and $S_i^{O2,T2}$ (red) are plotted against $S_i^{O2,T3}$.

6.2.2.2 Alteration of connections

In Figure 6.8, we observe that, as one could expect, the randomness in the response of the network to the stimuli increases with the percentage of edges' weights altered. However, one can still detect neurons that respond strongly to only one object's stimuli.

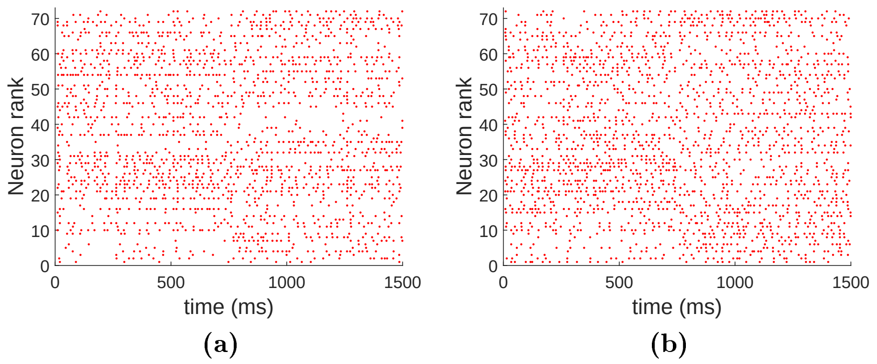
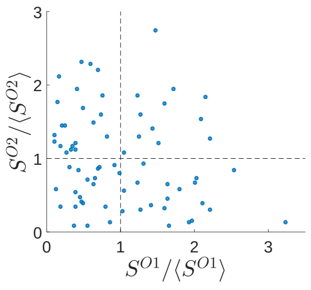


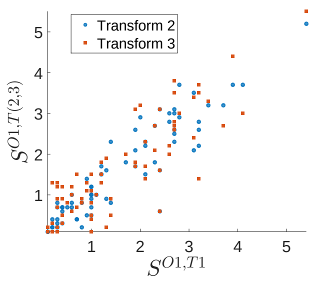
Figure 6.8: (a) Raster plot of the trained network where 30% of the connections were randomly altered. (b) Raster plot of the trained network where 70% of the connections were randomly modified. The horizontal axis gives the time; the vertical axis gives the rank of the neurons. The red dots represent the spikes of the neurons.

In the panel of Figures 6.9, two observations can be made. First, Figures 6.9.a and 6.9.d show that, in contrast to the previous case, the differentiation between the objects decreases as we increase the percentage of connections modified. This remark is supported by the increased randomness observed on the raster plots in Figure 6.8. However, in Figures 6.9.b, 6.9.c, 6.9.e, and 6.9.f we observe still a correlation between the transforms, which implies that the network responds the same to the different stimuli associated with an object.

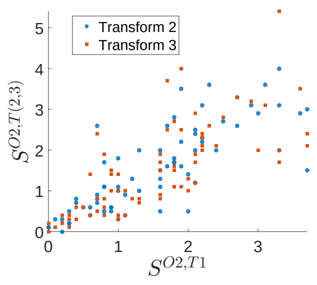
The discrimination ratio distributions in Figure 6.10 highlights the robustness of the performance of the system to this source of damage. The network maintains performance up to 50% of damage before dropping below the level reached after 2 sequences of training.



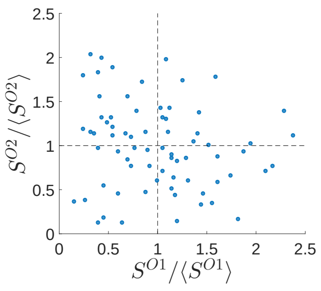
(a)



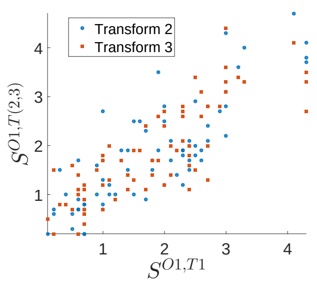
(b)



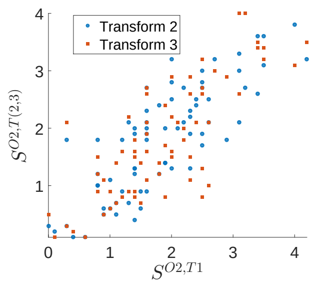
(c)



(d)



(e)



(f)

Figure 6.9: These figures correspond to scatter plots. (a), (b), and (c) corresponds to the trained network with 30% of connections modified, and (d), (e), and (f) are associated with the trained network with 70% connections modified. (a) and (d) are the scatter plots of the mean spike counts for each cell S_i^{O1} andS_i^{O2} corresponding to responses of the network to object 1 and object 2, respectively. These graphs are normalised with the mean value over all cells for each object $\langle S^{O1}\rangle$ and $\langle S^{O2}\rangle$, which implies that the value 1 on both axis represents the mean response over all cells to each object. (b) and (e) are the scatter plots of the mean spike counts for each cell $S_i^{O1,T1}$, $S_i^{O1,T2}$,and $S_i^{O1,T3}$ corresponding to responses of the network to each transform associated with object 1. $S_i^{O1,T1}$ (blue) and $S_i^{O1,T2}$ (red) are plotted against $S_i^{O1,T3}$. (c) and (f) are the scatter plots of the mean spike counts for each cell $S_i^{O2,T1}$, $S_i^{O2,T2}$,and $S_i^{O2,T3}$ corresponding to responses of the network to each transform associated with object 2. $S_i^{O2,T1}$ (blue) and $S_i^{O2,T2}$ (red) are plotted against $S_i^{O2,T3}$.

This type of damage affects the network differently when compared to the simple removal of connexions. In contrast, we observed the capacity of the neuron in the network to differentiate between objects was primarily affected. This is consistent with an homogenisation of the firing across the E2 layer. It could be interpreted as the network having relatively the same response to each stimulus and thus trouble differentiating object. Note, however, that this become sensible when the percentage of connections altered exceeds 50%. Hence, the network shows robustness to this model of damage as well.

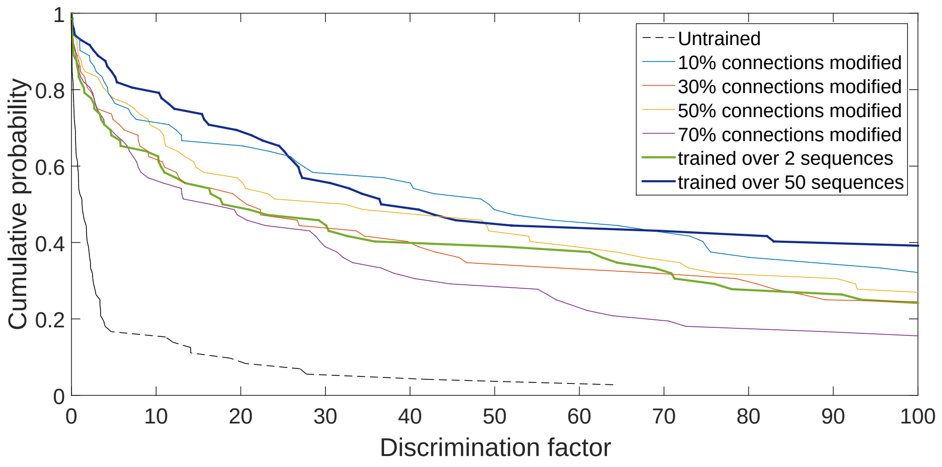


Figure 6.10: Cumulative plot of the discrimination ratio distribution in the output excitatory layer. The vertical axis gives the probability that a neuron has a discrimination ratio above a certain value, which is given on the horizontal axis. The horizontal axis is cut at 100 to improve readability. The dashed line corresponds to the response of the untrained network. The thicker lines correspond to the trained network after 2 sequences (green) and after 50 sequences (blue). The other lines correspond to the trained network after changing randomly 10% (blue), 30% (red), 50% (orange), and 70% (purple) of the connections.

Chapter 7

Conclusion

In this project, we developed a toy model and we used it to investigate a way to achieve object recognition in a neuronal system and to study the impact of damage on connections in a system.

We used the leaky Integrate-and-Fire model to describe the behaviour of each neuron of the system. This model allows one to track the individual spikes of each neuron. This is necessary for the spike-dependent plasticity of the synapses on which the continuous transformation learning rule is based.

We used a multilayer network formalism to describe the architecture of the system and how components interact with each other. The network defined is composed of four layers, which are sorted into two clusters. The input cluster models the retina, and the output cluster represents a part of the primary visual cortex (V1).

We then successfully trained our toy network to recognise two different objects and their transforms and to differentiate between the objects (see Section 6.2.1). The transforms of an object can be interpreted as the stimuli produced by the object observed at different angles.

We simulated damage in the system using two different methods to affect the inter-excitatory-layer connections: (1) direct removal of a connection and (2) random change of the weight of a connection. The simulations on these deteriorated systems highlighted the robustness and stability of the trained network. Indeed, the network exhibits little performance drop even when the percentage of the connections affected exceeds 50%. Recall that performance is associated to good discrimination between different objects presented to the network. This entails cell specificity, which means that a cell performs well if it responds strongly to one object and weakly to the other. We evaluate specificity using both visual interpretation of graphs and the discrimination ratio computed with an ANOVA method.

Future work could entail training the system with either more objects or more complicated objects for instance, using actual pictures, as in [35]. We expect that the performance of a network will drop more rapidly with an increase in the percentage of damaged connections.. Another interesting direction would be to examine the impact on layers that receive inputs from the output cluster. We observed that the level of activity decreases with the percentage of damage, the layers located further away from the input cluster are expected to rapidly lose specificity. The notion of distance we have in mind here corresponds to the number of "inter-layers" between the input and the output considered. This hypothesis, if confirmed, will then lead to a similar question asked in [39]: how can the stability observed in real neuronal systems be explained? The answer proposed in their article is that the stability comes inherently from the network connectivities. One could then test how different type of connectivities between clusters impact both the learning phase and the process of stimuli by a damaged network.

Appendix A

Integrate-and-Fire code with learning

```
function [D_int_layer] = IF_CODE_inputconst(N,D,D_int_layer,tburst,...
nburst,dt,I,xi,flag_learning)

% N              -  Number of neurons in excitatory and inhibitory
%                   layers (2*1 vector)
% D              -  Fixed weights values (2*2 matrix)
% D_int_layer    -  Relative weight between E1 and E2 layer
% tburst         -  Time during which each stimulus is presented
% nburst         -  Overall number of stimulation (each stimulus is
%                   presented nburt/6 times
% dt             -  Fixed time-step
% I              -  Intensity amplitude received by each neuron during
%                   stimulation
% xi             -  White noise standard deviation
% flag_learning -  Boolean stating turning on/off the learning phase

DEBUG =1;
if nargin <1 && DEBUG==1
clc;
warning ('off','all');
```

```
N = [18 72]*1;
dt = 0.01;
tburst = 100;
nburst = 60;
D = [0.08 0.08 ; 0.03 0.01];
D(1,:) = D(1,:)*2/3;
D(2,:) = D(2,:)*14/3;
% Relative weights between E1 and E2
load connections;
D_int_layer = connections;
flag_learning = 1;

%Stimuli parameters
P = 28;% Number of neurons stimulated by one transform
Pt = 4;% Translation between successive transform
I = 0.1;% Intensity of stimulus

%Noise
xi = 0.01;
end

eps = 10^(-5);

%Learning rule parameters
D_i = 0.1*ones(N(2),N(2));
C_i = 0.1*ones(N(2),N(2));
alpha_D = 0.5; alpha_C = 0.5;

%Simulation max time
TMAX = tburst*nburst;

%Parameters
TAU_V=20;VT=1;VR=0;
NI1 = N(1); NE1 = N(2);% Number of inihbitory/excitatory neurons on 1st
    layer
```

```
NI2 = N(1); NE2 = N(2);% Number of inihbitory/excitatory neurons on 2nd
    layer

% DQR are coupling strengths from R to Q;
DII = D(1,1); DIE = D(1,2); DEI = D(2,1); DEE = D(2,2); %Synapses
    strengths intra layer

%Stimuli
IE1_1 = I* [ones(P,1);zeros(NE1-P,1)];
IE1_2 = I* [zeros(Pt,1);ones(P,1);zeros(NE1-P-Pt,1)];
IE1_3 = I* [zeros(2*Pt,1);ones(P,1);zeros(NE1-P-2*Pt,1)];
IE2_1 = I* [zeros(NE1-P,1);ones(P,1)];
IE2_2 = I* [zeros(NE1-P-Pt,1);ones(P,1);zeros(Pt,1)];
IE2_3 = I* [zeros(NE1-P-2*Pt,1);ones(P,1);zeros(2*Pt,1)];

%Potential initialisation
VE1 = zeros(NE1,1);
VI1 = zeros(NI1,1);
VE2 = zeros(NE2,1);
VI2 = zeros(NI2,1);

% Stocking structures
time_serie = [];
E1_fired_serie = {}; E2_fired_serie = {}; I1_fired_serie = {};
    I2_fired_serie = {};
E2_fire_count = zeros(NE2,1);
E2_spike1 = zeros(NE2, nburst/2);
E2_spike2 = zeros(NE2, nburst/2);
iteration=0; t_sum=0;
dtmax= dt;

% Structures keeping track of refractory period for each neuron
Tref = 2;% Refractory time
TR_E1 = zeros(NE1,1); TR_I1 = zeros(NI1,1); TR_E2 = zeros(NE2,1); TR_I2
    = zeros(NI2,1);
```

```
while (t_sum<TMAX);
if (mod(floor(t_sum/tburst),6)== 0)% First stimulus
dt_temp = dtmax;
if (any(TR_E1>eps) || any(TR_I1>eps) || any(TR_E2>eps) ||
    any(TR_I2>eps))
% Find the next neuron getting out of the refractory period (if
% any)
dt_temp = min([ min(TR_E1(TR_E1>eps)), min(TR_I1(TR_I1>eps)),...
min(TR_E2(TR_E2>eps)), min(TR_I2(TR_I2>eps))]);
end
% Set time step to the minimum between the time the next neuron gets
% out the refractory period, the time left until the stimulus is
% changed and the fixed time step dtmax.
dt_cur = min([dtmax,((floor(t_sum/tburst)+1)*tburst - t_sum),dt_temp]);
t_sum = t_sum+dt_cur;% Update simulation time
% Update potentials and refractory times of neurons
edt = exp(-dt_cur/TAU_V);
VE1 = (VE1*edt + IE1_1*dt_cur + xi*randE1).*(TR_E1==0)+VR;
VI1 = (VI1*edt + xi*randI1).*(TR_I1==0)+VR;
VE2 = (VE2*edt+ xi*randE2).*(TR_E2==0)+VR;
VI2 = (VI2*edt+ xi*randI2).*(TR_I2==0)+VR;
TR_E1 = max(TR_E1 - dt_cur,0); TR_I1 = max(TR_I1 - dt_cur,0);
TR_E2 = max(TR_E2 - dt_cur,0); TR_I2 = max(TR_I2 - dt_cur,0);
if (any(VI1(VI1 >= VT)) || any(VE1(VE1 >= VT))|| ...
any(VI2(VI2 >= VT)) || any(VE2(VE2 >= VT)))
% Detect if any neurons crossed the threshold
[E1_fired,I1_fired,E2_fired, I2_fired,VE1,VI1,VE2,VI2,C_i,D_i,...
D_int_layer] = getMFE_ifdyn(VE1,VI1,VE2,VI2,D,D_int_layer,...
C_i,D_i,flag_learning); % Process the impact of the spike(s)
iteration=iteration+1;
% Reset potentials
VE1(E1_fired)=VR; VI1(I1_fired)=VR;
VE2(E2_fired)=VR; VI2(I2_fired)=VR;
```

```
% Set neurons refractory period
TR_E1(E1_fired) = Tref; TR_I1(I1_fired) = Tref;
TR_E2(E2_fired) = Tref; TR_I2(I2_fired) = Tref;
% Update structures that hold the spike counts
E2_fire_count(E2_fired) = E2_fire_count(E2_fired) + 1;
time_serie = [time_serie, t_sum];
E1_fired_serie{iteration} = E1_fired;
E2_fired_serie{iteration} = E2_fired;
I1_fired_serie{iteration} = I1_fired;
I2_fired_serie{iteration} = I2_fired;
end
if (mod(floor(t_sum/tburst),6)== 1)
% Reset potential if transform changes, and store number
% of spikes counted
VE1 = zeros(NE1,1);
VI1 = zeros(NI1,1);
VE2 = zeros(NE2,1);
VI2 = zeros(NI2,1);
E2_spike1(:,1 + (floor(t_sum/tburst/6))*3) = E2_fire_count;
E2_fire_count = zeros(NE2,1);
end

elseif (mod(floor(t_sum/tburst),6)== 1)% Second stimulus
dt_temp = dtmax;
if (any(TR_E1>eps) || any(TR_I1>eps) || any(TR_E2>eps) ||
    any(TR_I2>eps))
dt_temp = min([ min(TR_E1(TR_E1>eps)), min(TR_I1(TR_I1>eps)),...
min(TR_E2(TR_E2>eps)), min(TR_I2(TR_I2>eps))]);
end
dt_cur = min([dtmax,((floor(t_sum/tburst)+1)*tburst - t_sum),dt_temp]);
t_sum=t_sum+dt_cur; % update simulation time
edt = exp(-dt_cur/TAU_V);
VE1 = (VE1*edt + IE1_2*dt_cur + xi*(randn(NE1,1))).*(TR_E1==0)+VR;
VI1 = (VI1*edt + xi*(randn(NI1,1))).*(TR_I1==0)+VR;
VE2 = (VE2*edt+ xi*(randn(NE2,1))).*(TR_E2==0)+VR;
```

```
VI2 = (VI2*edt+ xi*(randn(NI2,1))).*(TR_I2==0)+VR;
TR_E1 = max(TR_E1 - dt_cur,0); TR_I1 = max(TR_I1 - dt_cur,0);
TR_E2 = max(TR_E2 - dt_cur,0); TR_I2 = max(TR_I2 - dt_cur,0);
if (any(VI1(VI1 >= VT)) || any(VE1(VE1 >= VT))||...
any(VI2(VI2 >= VT)) || any(VE2(VE2 >= VT)))
[E1_fired,I1_fired,E2_fired, I2_fired, VE1,VI1,VE2,VI2,C_i,D_i,...
D_int_layer] = getMFE_ifdyn(VE1,VI1,VE2,VI2,D,D_int_layer,...
C_i,D_i,alpha_C,alpha_D);
iteration=iteration+1;
VE1(E1_fired)=VR; VI1(I1_fired)=VR;
VE2(E2_fired)=VR; VI2(I2_fired)=VR;
TR_E1(E1_fired) = Tref; TR_I1(I1_fired) = Tref;
TR_E2(E2_fired) = Tref; TR_I2(I2_fired) = Tref;
time_serie = [time_serie, t_sum];
E1_fired_serie{iteration} = E1_fired;
E2_fired_serie{iteration} = E2_fired;
I1_fired_serie{iteration} = I1_fired;
I2_fired_serie{iteration} = I2_fired;
E2_fire_count(E2_fired) = E2_fire_count(E2_fired) + 1;
end
if (mod(floor(t_sum/tburst),6)== 2)
VE1 = zeros(NE1,1);
VI1 = zeros(NI1,1);
VE2 = zeros(NE2,1);
VI2 = zeros(NI2,1);
E2_spike1(:,2 + (floor(t_sum/tburst/6))*3) = E2_fire_count;
E2_fire_count = zeros(NE2,1);
end;

elseif (mod(floor(t_sum/tburst),6)== 2)% Third stimulus
dt_temp = dtmax;
if (any(TR_E1>eps) || any(TR_I1>eps) || any(TR_E2>eps) ||
    any(TR_I2>eps))
dt_temp = min([ min(TR_E1(TR_E1>eps)), min(TR_I1(TR_I1>eps)),...
min(TR_E2(TR_E2>eps)), min(TR_I2(TR_I2>eps))]);
```

```
end
dt_cur = min([dtmax,((floor(t_sum/tburst)+1)*tburst - t_sum),dt_temp]);
t_sum=t_sum+dt_cur;
edt = exp(-dt_cur/TAU_V);
VE1 = (VE1*edt + IE1_3*dt_cur + xi*(randn(NE1,1))).*(TR_E1==0)+VR;
VI1 = (VI1*edt + xi*(randn(NI1,1))).*(TR_I1==0)+VR;
VE2 = (VE2*edt+ xi*(randn(NE2,1))).*(TR_E2==0)+VR;
VI2 = (VI2*edt+ xi*(randn(NI2,1))).*(TR_I2==0)+VR;
TR_E1 = max(TR_E1 - dt_cur,0); TR_I1 = max(TR_I1 - dt_cur,0);
TR_E2 = max(TR_E2 - dt_cur,0); TR_I2 = max(TR_I2 - dt_cur,0);
if (any(VI1(VI1 >= VT)) || any(VE1(VE1 >= VT))||...
any(VI2(VI2 >= VT)) || any(VE2(VE2 >= VT)))
[E1_fired,I1_fired,E2_fired, I2_fired, VE1,VI1,VE2,VI2,C_i,...
D_i,D_int_layer] = getMFE_ifdyn(VE1,VI1,VE2,VI2,D,D_int_layer,...
C_i,D_i,alpha_C,alpha_D);
iteration=iteration+1; dt_cur=0;t_sum=t_sum+dt_cur;
VE1(E1_fired)=VR; VI1(I1_fired)=VR;
VE2(E2_fired)=VR; VI2(I2_fired)=VR;
TR_E1(E1_fired) = Tref; TR_I1(I1_fired) = Tref;
TR_E2(E2_fired) = Tref; TR_I2(I2_fired) = Tref;
time_serie = [time_serie, t_sum];
E1_fired_serie{iteration} = E1_fired;
E2_fired_serie{iteration} = E2_fired;
I1_fired_serie{iteration} = I1_fired;
I2_fired_serie{iteration} = I2_fired;
E2_fire_count(E2_fired) = E2_fire_count(E2_fired) + 1;
end
if (mod(floor(t_sum/tburst),6)== 3)
VE1 = zeros(NE1,1);
VI1 = zeros(NI1,1);
VE2 = zeros(NE2,1);
VI2 = zeros(NI2,1);
E2_spike1(:,3 + (floor(t_sum/tburst/6))*3) = E2_fire_count;
E2_fire_count = zeros(NE2,1);
end
```

```
elseif (mod(floor(t_sum/tburst),6)== 3)% Fourth stimulus
dt_temp = dtmax;
if (any(TR_E1>eps) || any(TR_I1>eps) || any(TR_E2>eps) ||
    any(TR_I2>eps))
dt_temp = min([ min(TR_E1(TR_E1>eps)), min(TR_I1(TR_I1>eps)),...
min(TR_E2(TR_E2>eps)), min(TR_I2(TR_I2>eps))]);
end
dt_cur = min([dtmax,((floor(t_sum/tburst)+1)*tburst - t_sum),dt_temp]);
t_sum=t_sum+dt_cur;
edt = exp(-dt_cur/TAU_V);
VE1 = (VE1*edt + IE2_1*dt_cur + xi*(randn(NE1,1))).*(TR_E1==0)+VR;
VI1 = (VI1*edt + xi*(randn(NI1,1))).*(TR_I1==0)+VR;
VE2 = (VE2*edt+ xi*(randn(NE2,1))).*(TR_E2==0)+VR;
VI2 = (VI2*edt+ xi*(randn(NI2,1))).*(TR_I2==0)+VR;
TR_E1 = max(TR_E1 - dt_cur,0); TR_I1 = max(TR_I1 - dt_cur,0);
TR_E2 = max(TR_E2 - dt_cur,0); TR_I2 = max(TR_I2 - dt_cur,0);
if (any(VI1(VI1 >= VT)) || any(VE1(VE1 >= VT))||...
any(VI2(VI2 >= VT)) || any(VE2(VE2 >= VT)))
[E1_fired,I1_fired,E2_fired, I2_fired, VE1,VI1,VE2,VI2,C_i,...
D_i,D_int_layer] = getMFE_ifdyn(VE1,VI1,VE2,VI2,D,D_int_layer,...
C_i,D_i,alpha_C,alpha_D);
iteration=iteration+1; dt_cur=0;t_sum=t_sum+dt_cur;
VE1(E1_fired)=VR; VI1(I1_fired)=VR;
VE2(E2_fired)=VR; VI2(I2_fired)=VR;
TR_E1(E1_fired) = Tref; TR_I1(I1_fired) = Tref;
TR_E2(E2_fired) = Tref; TR_I2(I2_fired) = Tref;
time_serie = [time_serie, t_sum];
E1_fired_serie{iteration} = E1_fired;
E2_fired_serie{iteration} = E2_fired;
I1_fired_serie{iteration} = I1_fired;
I2_fired_serie{iteration} = I2_fired;
E2_fire_count(E2_fired) = E2_fire_count(E2_fired) + 1;
end
if (mod(floor(t_sum/tburst),6)== 4)
```

```
VE1 = zeros(NE1,1);
VI1 = zeros(NI1,1);
VE2 = zeros(NE2,1);
VI2 = zeros(NI2,1);
E2_spike2(:,1+floor(t_sum/tburst/6)*3) = E2_fire_count;
E2_fire_count = zeros(NE2,1);
end

elseif (mod(floor(t_sum/tburst),6)== 4)% Fifth stimulus
dt_temp = dtmax;
if (any(TR_E1>eps) || any(TR_I1>eps) || any(TR_E2>eps) ||
    any(TR_I2>eps))
dt_temp = min([ min(TR_E1(TR_E1>eps)), min(TR_I1(TR_I1>eps)),...
min(TR_E2(TR_E2>eps)), min(TR_I2(TR_I2>eps))]);
end
dt_cur = min([dtmax,((floor(t_sum/tburst)+1)*tburst - t_sum),dt_temp]);
t_sum=t_sum+dt_cur;
edt = exp(-dt_cur/TAU_V);
VE1 = (VE1*edt + IE2_2*dt_cur + xi*(randn(NE1,1))).*(TR_E1==0)+VR;
VI1 = (VI1*edt + xi*(randn(NI1,1))).*(TR_I1==0)+VR;
VE2 = (VE2*edt+ xi*(randn(NE2,1))).*(TR_E2==0)+VR;
VI2 = (VI2*edt+ xi*(randn(NI2,1))).*(TR_I2==0)+VR;
TR_E1 = max(TR_E1 - dt_cur,0); TR_I1 = max(TR_I1 - dt_cur,0);
TR_E2 = max(TR_E2 - dt_cur,0); TR_I2 = max(TR_I2 - dt_cur,0);
if (any(VI1(VI1 >= VT)) || any(VE1(VE1 >= VT))||...
any(VI2(VI2 >= VT)) || any(VE2(VE2 >= VT)))
[E1_fired,I1_fired,E2_fired, I2_fired, VE1,VI1,VE2,VI2,C_i,...
D_i,D_int_layer] = getMFE_ifdyn(VE1,VI1,VE2,VI2,D,D_int_layer,...
C_i,D_i,alpha_C,alpha_D);
iteration=iteration+1; dt_cur=0;t_sum=t_sum+dt_cur;
VE1(E1_fired)=VR; VI1(I1_fired)=VR;
VE2(E2_fired)=VR; VI2(I2_fired)=VR;
TR_E1(E1_fired) = Tref; TR_I1(I1_fired) = Tref;
TR_E2(E2_fired) = Tref; TR_I2(I2_fired) = Tref;
time_serie = [time_serie, t_sum];
```

```
E1_fired_serie{iteration} = E1_fired;
E2_fired_serie{iteration} = E2_fired;
I1_fired_serie{iteration} = I1_fired;
I2_fired_serie{iteration} = I2_fired;
E2_fire_count(E2_fired) = E2_fire_count(E2_fired) + 1;
end
if (mod(floor(t_sum/tburst),6)== 5)
VE1 = zeros(NE1,1);
VI1 = zeros(NI1,1);
VE2 = zeros(NE2,1);
VI2 = zeros(NI2,1);
E2_spike2(:,2 + (floor(t_sum/tburst/6))*3) = E2_fire_count;
E2_fire_count = zeros(NE2,1);
end;

else %if (mod(floor(t_sum/tburst),6)== 5) % Sixth stimulus
dt_temp = dtmax;
if (any(TR_E1>eps) || any(TR_I1>eps) || any(TR_E2>eps) ||
    any(TR_I2>eps))
dt_temp = min([ min(TR_E1(TR_E1>eps)), min(TR_I1(TR_I1>eps)),...
min(TR_E2(TR_E2>eps)), min(TR_I2(TR_I2>eps))]);
end
dt_cur = min([dtmax,((floor(t_sum/tburst)+1)*tburst - t_sum),dt_temp]);
t_sum=t_sum+dt_cur;
edt = exp(-dt_cur/TAU_V);
VE1 = (VE1*edt + IE2_3*dt_cur + xi*(randn(NE1,1))).*(TR_E1==0)+VR;
VI1 = (VI1*edt + xi*(randn(NI1,1))).*(TR_I1==0)+VR;
VE2 = (VE2*edt+ xi*(randn(NE2,1))).*(TR_E2==0)+VR;
VI2 = (VI2*edt+ xi*(randn(NI2,1))).*(TR_I2==0)+VR;
TR_E1 = max(TR_E1 - dt_cur,0); TR_I1 = max(TR_I1 - dt_cur,0);
TR_E2 = max(TR_E2 - dt_cur,0); TR_I2 = max(TR_I2 - dt_cur,0);
if (any(VI1(VI1 >= VT)) || any(VE1(VE1 >= VT))||...
any(VI2(VI2 >= VT)) || any(VE2(VE2 >= VT)))
[E1_fired,I1_fired,E2_fired, I2_fired, VE1,VI1,VE2,VI2,C_i,...
D_i,D_int_layer] = getMFE_ifdyn(VE1,VI1,VE2,VI2,D,D_int_layer,...
C_i,D_i,alpha_C,alpha_D);
iteration=iteration+1; dt_cur=0;t_sum=t_sum+dt_cur;
VE1(E1_fired)=VR; VI1(I1_fired)=VR;
VE2(E2_fired)=VR; VI2(I2_fired)=VR;
TR_E1(E1_fired) = Tref; TR_I1(I1_fired) = Tref;
TR_E2(E2_fired) = Tref; TR_I2(I2_fired) = Tref;
time_serie = [time_serie, t_sum];
E1_fired_serie{iteration} = E1_fired;
E2_fired_serie{iteration} = E2_fired;
I1_fired_serie{iteration} = I1_fired;
I2_fired_serie{iteration} = I2_fired;
E2_fire_count(E2_fired) = E2_fire_count(E2_fired) + 1;
end
if (mod(floor(t_sum/tburst),6)== 0)
VE1 = zeros(NE1,1);
VI1 = zeros(NI1,1);
VE2 = zeros(NE2,1);
VI2 = zeros(NI2,1);
E2_spike2(:,3 + (floor(t_sum/tburst/6)-1)*3) = E2_fire_count;
E2_fire_count = zeros(NE2,1);
end;
end
end;%while (t_sum<TMAX);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function
    [E1_fired,I1_fired,E2_fired,I2_fired,VE1,VI1,VE2,VI2,C_i,D_i,...
D_int_layer] = getMFE_ifdyn(VE1,VI1,VE2,VI2,D,D_int_layer,...
C_i,D_i,flag_learning)
% This function deals with the infinitesimal-time system
TAU_V=20;VT=1;VR=0;
tau_delta = 10;
DII = D(1,1); DIE = D(1,2);
DE12 = D(2,2);DEI = D(2,1);
```

```
% first population of inihbitory neurons
I1_fired = find(VI1>=VT);
LI1 = length(I1_fired);
I1_remaining = find(VI1<VT);
if LI1>0;
VE1 = VE1 - DEI*LI1;
VI1(I1_remaining) = VI1(I1_remaining) - DII*LI1;
end;%if LI>0;

% second population of inihbitory neurons
I2_fired = find(VI2>=VT);
LI2 = length(I2_fired);
I2_remaining = find(VI2<VT);
if LI2>0;
VE2 = VE2 - DEI*LI2;
VI2(I2_remaining) = VI2(I2_remaining) - DII*LI2;
end;%if LI2>0;

% first layer excitatory neurons
E1_fired = find(VE1>=VT);
LE1 = length(E1_fired);
E1_remaining = find(VE1<VT);
E2_remaining = find(VE2<VT);
% Feed forward to 2nd layer
VE2(E2_remaining) = VE2(E2_remaining) ...
+ (DE12*14/3)*sum(D_int_layer(E2_remaining,E1_fired),2);
total_V_to_add_to_I1 = DIE*LE1;
while (total_V_to_add_to_I1>0);
possible_I1_spikes = find(VI1(I1_remaining)>=VT-total_V_to_add_to_I1);
[max_I1,ind_I1] = max(VI1(I1_remaining)); ind_I1 = I1_remaining(ind_I1);
if (isempty(possible_I1_spikes));
V_to_add_to_I1 = total_V_to_add_to_I1;
VI1(I1_remaining) = VI1(I1_remaining) + V_to_add_to_I1;
total_V_to_add_to_I1 = 0;
```

```
else %(~isempty(possible_I1_spikes));
V_to_add_to_I1 = VT - max_I1;
VE1(E1_remaining) = VE1(E1_remaining) - DEI;
I1_fired = [I1_fired(:);ind_I1];
I1_remaining = setdiff(I1_remaining,ind_I1); LI1=LI1+1;
VI1(I1_remaining) = VI1(I1_remaining) - DII + V_to_add_to_I1;
total_V_to_add_to_I1 = total_V_to_add_to_I1 - V_to_add_to_I1;
end;%if any possible spikes;
end;%while (total_V_to_add>0);


if flag_learning % If learning turned on, deals with pre-synaptic spikes
C_i(:,E1_fired) = C_i(:,E1_fired) + alpha_C*(1-C_i(:,E1_fired));
Dint(:,E1_fired) = Dint(:,E1_fired) ...
- repmat(D_i,1,length(E1_fired)).*D_int_layer(:,E1_fired)/tau_delta;
end

% Second layer excitatory neurons
E2_fired = find(VE2>=VT);
LE2 = length(E2_fired);
E2_remaining = find(VE2<VT);
total_V_to_add_to_I2 = DIE*LE2;
while (total_V_to_add_to_I2>0);
possible_I2_spikes = find(VI2(I2_remaining)>=VT-total_V_to_add_to_I2);
[max_I2,ind_I2] = max(VI2(I2_remaining)); ind_I2 = I2_remaining(ind_I2);
if isempty(possible_I2_spikes)
V_to_add_to_I2 = total_V_to_add_to_I2;
VI2(I2_remaining) = VI2(I2_remaining) + V_to_add_to_I2;
total_V_to_add_to_I2 = 0;
else %~isempty(possible_I2_spikes)
V_to_add_to_I2 = VT - max_I2;
VE2(E2_remaining) = VE2(E2_remaining) - DEI;
I2_fired = [I2_fired(:);ind_I2];
I2_remaining = setdiff(I2_remaining,ind_I2); LI2=LI2+1;
VI2(I2_remaining) = VI2(I2_remaining) - DII + V_to_add_to_I2;
```

```
total_V_to_add_to_I2 = total_V_to_add_to_I2 - V_to_add_to_I2;
end;%if any possible spikes;
end;%while (total_V_to_add>0);

if flag_learning % If learning turned on, deals with post-synaptic
    spikes
D_i(E2_fired) = D_i(E2_fired) + alpha_D*(1-D_i(E2_fired));
Dint(E2_fired,:) = Dint(E2_fired,:) ...
+ (1-Dint(E2_fired,:)).*C_i(E2_fired,:)/tau_delta;
end;%flag_learning
```

References

[1] Chris 73, Diberri, and TiZom. Licensed under CC BY-SA 3.0 via Commons - `https://commons.wikimedia.org/wiki/File:Action_potential.svg`, 2007.

[2] Peter Ashwin, Stephen Coombes, and Rachel Nicks. Mathematical frameworks for oscillatory network dynamics in neuroscience. arXiv: 150, 2015.

[3] Ernest Barreto, Brian Hunt, Edward Ott, and Paul So. Synchronization in networks of networks: The onset of coherent collective behavior in systems of interacting populations of heterogeneous oscillators. *Physical Review E*, 77(3):036107, 2008.

[4] Stefano Boccaletti, Ginestra Bianconi, Regino C. Herrero, Charo I. del Genio, Jesus Gómez-Gardeñes, Miguel Romance, Irene Sendiña Nadal, Zhen Wang, and Massimiliano Zanin. The structure and dynamics of multilayer networks. *Physics Reports*, 544(1):1–122, 2014.

[5] Stephen Coombes, Ruediger Thul, and Kyle C. A. Wedgwood. Nonsmooth dynamics in spiking neuron models. *Physica D*, 241(22):2042–2057, 2012.

[6] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

[7] Arnaud Delorme, Jacques Gautrais, Rufin Van Rullen, and Simon Thorpe. SpikeNET: a simulator for modeling large networks of integrate and fire neurons. *Neurocomputing*, 40(1999):38–40, 2001.

[8] Michiel D'Haene, Michiel Hermans, and Benjamin Schrauwen. Toward unified hybrid simulation techniques for spiking neural networks. *Neural Computation*, 26(6):1055–79, 2014.

[9] Michiel D'Haene, Benjamin Schrauwen, Jan Van Campenhout, and Dirk Stroobandt. Accelerating event-driven simulation of spiking neurons with multiple synaptic time constants. *Neural Computation*, 21(4):1068–1099, 2009.

[10] Rodney J. Douglas and Kevan A. C. Martin. Neuronal circuits of the neocortex. *Annual Review of Neuroscience*, 27:419–451, 2004.

[11] G. Bard Ermentrout and Nancy Kopell. Parabolic Bursting in an Excitable System Coupled with a Slow Oscillation. *SIAM Journal on Applied Mathematics*, 46(2):233–253, 1986.

[12] G. Bard Ermentrout and David H. Terman. *Mathematical Foundations of Neuroscience*. Springer Science & Business Media, 2010.

[13] Benjamin D. Evans and Simon M. Stringer. Transformation-invariant visual representations in self-organizing spiking neural networks. *Frontiers in Computational Neuroscience*, 6:46, 2012.

[14] Donald O. Hebb. *The Organization of Behavior: A Neuropshychology Theory*. Psychology Press, 2005.

[15] Judith A. Hirsch. Synaptic physiology and receptive field structure in the early visual pathway of the cat. *Cerebral Cortex*, 13(1):63–69, 2003.

[16] Alan L. Hodgkin. The local electric changes associated with repetitive action in a non-medullated axon. *The Journal of Physiology*, 107(2):165–181, 1948.

[17] Alan L. Hodgkin and Andrew F. Huxley. A quantitative description of membrane current and application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, 1952.

[18] Christopher J. Honey, Rolf Kotter, Michael Breakspear, and Olaf Sporns. Network structure of cerebral cortex shapes functional connectivity on multiple time scales. *Proceedings of the National Academy of Sciences USA*, 104(24):10240–10245, 2007.

[19] Christopher J. Honey and Olaf Sporns. Dynamical consequences of lesions in cortical networks. *Human Brain Mapping*, 29(7):802–809, 2008.

[20] John J. Hopfield and Andreas V. Herz. Rapid local synchronization of action potentials: Toward computation with coupled integrate-and-fire neurons. *Poceedings of the National Academy of Sciences*, 92(July):6655–6662, 1995.

[21] David Howell. *Statistical Methods for Psychology.* Cengage Learning, 2012.

[22] Quasar Jarosz. Licensed under CC BY-SA 3.0 via Commons - `https://commons.wikimedia.org/wiki/File:Neuron_Hand-tuned.svg`, 2009.

[23] Bernard Katz and Ricardo Miledi. Further study of the role of calcium in synaptic transmission. *The Journal of Physiology*, 207(3):789–801, 1970.

[24] Bernard Katz and Ricardo Miledi. The effect of prolonged depolarization on synaptic transfer in the stellate ganglion of the squid. *The Journal of Physiology*, 216:503–512, 1971.

[25] Bernard Katz and Ricardo Miledi. The statistical nature of the acetycholine potential and its molecular components. *The Journal of Physiology*, 224(3):665–699, 1972.

[26] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason A. Porter. Multilayer Networks. *Journal of Complex Networks*, 2(3):203–271, 2014.

[27] Nancy J Kopell, Howard J Gritton, Miles a Whittington, and Mark a Kramer. Perspective Beyond the Connectome : The Dynome. *Neuron*, 83(6):1319–1328, 2014.

[28] Geneviève Leuba and Rudolf Kraftsik. Changes in volume, surface estimate, three-dimensional shape and total number of neurons of the human primary visual cortex from midgestation until old age. *Anatomy and Embryology*, 190(4):351–366, 1994.

[29] David J. MacKay. *Information Theory, Inference and Learning Algorithms.* Cambridge University Press, 2003.

[30] Timothée Masquelier, Etienne Hugues, Gustavo Deco, and Simon J. Thorpe. Oscillations, phase-of-firing coding, and spike timing-dependent plasticity: an efficient learning scheme. *The Journal of Neuroscience*, 29(43):13484–13493, 2009.

[31] David Mclaughlin, Robert Shapley, Michael Shelley, and Dingeman J. Wielaard. A neuronal network model of macaque primary visual cortex (V1): Orientation selectivity and dynamics in the input layer 4C\alpha. *Proceedings of the National Academy of Sciences USA*, 97(14):8087–8092, 2000.

[32] Abigail Morrison, Sirko Straube, Hans Ekkehard Plesser, and Markus Diesmann. Exact subthreshold integration with continuous spike times in discrete-time neural network simulations. *Neural Computation*, 19(1):47–79, 2007.

[33] Pamela Moses and Joan Stiles. The lesion methodology: Contrasting views from adult and child studies. *Developmental Psychobiology*, 40(3):266–277, 2002.

[34] Mark E. J. Newman. *Networks: An Introduction.* Oxford University Press, Oxford, 2010.

[35] Laurent Perrinet, Arnaud Delorme, Manuel Samuelides, and Simon J. Thorpe. Networks of integrate-and-fire neuron using rank order coding A: How to implement spike time dependent Hebbian plasticity. *Neurocomputing*, 38-40:817–822, 2001.

[36] Aaditya V. Rangan and David Cai. Fast numerical methods for simulating large-scale integrate-and-fire neuronal networks. *Journal of Computational Neuroscience*, 22(1):81–100, 2007.

[37] Aaditya V. Rangan and Lai-Sang Young. Dynamics of spiking neurons: Between homogeneity and synchrony. *Journal of Computational Neuroscience*, 34(3):433–460, 2013.

[38] Aaditya V Rangan and Lai-Sang Young. Emergent dynamics in a model of visual cortex. *Journal of Computational Neuroscience*, 35(2):155–67, 2013.

[39] Saulo D. S. Reis, Yanqing Hu, Andrés Babino, José S. Andrade Jr, Santiago Canals, Mariano Sigman, and Hernán a. Makse. Avoiding catastrophic failure in correlated networks of networks. *Nature Physics*, 10, 2014.

[40] Edmund T. Rolls and Alessandro Treves. *Neural Networks and Brain Function.* Oxford University Press, Oxford, 1998.

[41] Eduardo Ros, Richard Carrillo, Eva M. Ortigosa, Boris Barbour, and Rodrigo Agís. Event-driven simulation scheme for spiking neural networks using lookup tables to characterize neuronal dynamics. *Neural Computation*, 18(12):2959–2993, 2006.

[42] Carla J. Schatz. The developing brain. *Scientific American*, 267(3):60–67, 1992.

[43] Paul So, Bernard C Cotton, and Ernest Barreto. Synchronization in interacting populations of heterogeneous oscillators with time-varying coupling. *Chaos*, 18(3):037114, 2008.

[44] Thomas Splettstoesser. Licensed under CC BY-SA 4.0 via Commons - `https://commons.wikimedia.org/wiki/File:SynapseSchematic_en.svg#/media/File:SynapseSchematic_en.svgitle`, 2015.

[45] Olaf Sporns. *Networks of the Brain.* MIT Press, 2011.

[46] Simon M. Stringer, G. Perry, Edmund. T. Rolls, and J. H. Proske. Learning invariant object recognition in the visual system with continuous transformations. *Biological Cybernetics*, 94(2):128–142, 2006.

[47] Todd W. Troyer, Anton E. Krukowski, Nicholas J. Priebe, and K. D. Miller. Contrast-invariant orientation tuning in cat visual cortex: thalamocortical input tuning and correlation-based intracortical connectivity. *The Journal of Neuroscience*, 18(15):5908–5927, 1998.

[48] Guy Wallis and Edmund T. Rolls. Invariant face and object recognition in the visual system. *Progress in Neurobiology*, 51(2):167–194, 1997.

[49] Daniel Walters, Simon Stringer, and Edmund Rolls. Path integration of head direction: updating a packet of neural activity at the correct speed using axonal conduction delays. *PLOS ONE*, 8(3), 2013.

[50] Lloyd Watts. Event-Driven Simulation of Networks of Spiking Neurons. In *Proceedings of the Sixth Neural Information Processing Systems Conference*, pages 927–934, San Francisco, 1993. Morgan Kaufmann.

[51] Robert W. Williams and Karl Herrup. The control of neuron number. *Annual Review of Neuroscience*, 11:423–453, 1988.

[52] Jiwei Zhang, Douglas Zhou, David Cai, and Aaditya V. Rangan. A coarse-grained framework for spiking neuronal networks: Between homogeneity and synchrony. *Journal of Computational Neuroscience*, 37(1):81–104, 2014.

[53] Jiwei W. Zhang and Aaditya V. Rangan. A reduction for spiking integrate-and-fire network dynamics ranging from homogeneity to synchrony. *Journal of Computational Neuroscience*, 38(2):355–404, 2015.

[54] Changsong Zhou, Lucia Zemanová, Gorka Zamora, Claus C. Hilgetag, and Jürgen Kurths. Hierarchical Organization Unveiled by Functional Connectivity in Complex Brain Networks. *Physical Review Letters*, 97(23):238103, 2006.

[55] Changsong Zhou, Lucia Zemanová, Gorka Zamora-López, Claus C Hilgetag, and Jürgen Kurths. Structure–function relationship in complex brain networks expressed by hierarchical synchronization. *New Journal of Physics*, 9(6):178–178, 2007.