



Norwegian University of
Science and Technology

Fuzzy Oscillations

a Novel Model for Solving Pattern Segmentation

Lester Johan Solbakken

Master of Science in Computer Science

Submission date: August 2009

Supervisor: Keith Downing, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

Study population models of spiking neural networks: how oscillations occur through degrees of synchrony between excitatory and inhibitory neurons. Create an abstract model that captures these dynamics, and study and discuss how networks of oscillatory nodes can provide an alternative to traditional neural networks and what their advantage would be.

Assignment given: 08. December 2008
Supervisor: Keith Downing, IDI

Fuzzy oscillations: a novel model for solving pattern segmentation

Lester Solbakken
IDI, NTNU
solbakke@idi.ntnu.no

Page intentionally left blank

Abstract

In this thesis we develop a novel network model that extends the traditional artificial neural network (ANN) model to include oscillatory behaviour. This model is able to correctly classify combinations of previously learned input patterns by grouping features that belong to the same category. This grouping process is termed segmentation and we show how synchrony of oscillations is the necessary missing component of ANNs to be able to perform this segmentation.

Using this model we go on to show that top-down modulatory feedback is necessary to enable separation of multiple objects in a scene and segmentation of their individual features. This type of feedback is distinctly different than recurrency and is what enables the rich dynamics between the nodes of our network.

Additionally, we show how our model's dynamics avoid the combinatorial explosion in required training repetitions of traditional feed-forward classification networks. In these networks, relations between objects must explicitly be learned. In contrast, the dynamics of modulatory feedback allow us to defer calculation of these relations until run-time, thus creating a more robust system.

We call our model *Fuzzy Oscillations*, and it achieves good results when compared to existing models. However, oscillatory neural network models successful in achieving segmentation are a relatively recent development. We thus feel that our model is a contribution to the field of oscillatory neural networks.

Page intentionally left blank

Contents

1	Introduction	1
2	Background	4
2.1	The binding problem	4
2.2	Populations and oscillations	10
2.3	The importance of feedback	15
2.4	Existing computational models	18
3	Methods and implementation	23
3.1	Populations of spiking neurons	23
3.2	Analyzing population activity	26
3.3	Populations as systems of oscillators	29
3.4	Partial synchrony	33
3.5	The model	36
4	Experiments and results	45
4.1	Model fundamentals	46
4.2	Simple image recognition	49
4.3	Performance comparison	56
5	Discussion	62
5.1	Model details	62
5.2	Learning	66
5.3	Connection to cognitive processes	67
6	Conclusion and future work	71
A	Example code	74

List of Figures

2.1	The binding problem	5
2.2	Artificial neural networks	6
2.3	Spiking neurons and STDP	8
2.4	Synchronization of spiking neurons	9
2.5	Systems of oscillators	13
2.6	Levels of processing in the brain	14
2.7	Simplified cortical connections	16
3.1	Izhikevich network at various stages in STDP training	25
3.2	Neural population as a system of many oscillators	26
3.3	Local field potentials at various stages of network evolution	27
3.4	Power spectrum at various stages of network evolution	29
3.5	General oscillator	30
3.6	Power spectrums of systems of oscillators	31
3.7	Addition of oscillators	32
3.8	Lorentzian influence	35
3.9	Lorentzian functions and resonance	37
3.10	Feed-forward and feedback network activity	38
3.11	Simple feed-forward and feedback dynamic	41
4.1	Simple P-R scenario	47
4.2	Wheels-Barbell-Chassis scenario	49
4.3	Input images for simple recognition	50
4.4	Composited patterns with noise	52
4.5	Separation and segmentation accuracy in the presence of noise	54
4.6	Segmentation in an occluded scene	55
4.7	Feature detection of letter patterns	57
4.8	Multiple simultaneous letter patterns	58
4.9	C-D example	59
5.1	Equal influence distance as a function of connected nodes	63
5.2	Non-linearity in activation	65
5.3	Rubin's vase	68
5.4	Recognition loop	69

Chapter 1

Introduction

About 5 decades ago, the progress within the newly established field of Artificial Intelligence was astonishing. Computer programs were written that achieved great success at high-level thinking, such as planning, reasoning, and even communicating in natural languages. The optimism was high, and the founders of the field predicted that machines would have the general level of intelligence of an average human being within a decade or two. Indeed, such was the optimism of achieving AI that the human brain was usually described in terms of computer architecture.

About 2 decades ago, it was becoming clear that the original approach to AI, now termed as *symbolic* AI, would never be able to imitate all the processes of human cognition, such as perception and learning. A new kind of AI, called *sub-symbolic* AI emerged, and a new set of computer programs were written that achieved great success at some of these these lower-level processes. The difference lies in whereas symbolic AI can be compared with a cognitive psychological, top-down, approach to understanding the mind, sub-symbolic AI is a bottom-up approach. Indeed, sub-symbolic AI was inspired by neuroscience that seeks to understand the functioning of the brain by studying its smallest components.

As knowledge about the brain increased, the metaphor of the brain as a computational mechanism slowly changed. With the advent of sub-symbolic methods such as artificial neural networks, it has become of great interest to understand the processes in the brain to inspire advances in computation and engineering. In a sense, the tables have turned. Artificial neural networks are now found in a huge number of different applications: pattern recognition such as face identification and radar systems, data mining for knowledge discovery, medical diagnosis and even automated trading systems in finance. Processes used by the brain to solve daily problems can be applied to a large number of different

tasks. So while the field of neuroscience might ultimately be more interested in medical applications such as curing neurological disorders, it is also a potential gold mine for possible engineering applications.

Walter Freeman points out in [15] that the successful simulation of animal intelligence could lead to a new kind of artificial intelligence, for example robots that could operate autonomously in unfamiliar environments. The critical ability would be to evaluate local conditions and make decisions to optimize performance without the need for immediate and continuous supervision. While the state of the art is not quite there yet, there are still burning questions of how an organism perceives the world around it to efficiently make sense of its environment. For instance, to survive animals must interpret their visual input rapidly. If encircled by a predator pack it helps to identify the best escape route quickly, as does identification of food while moving around in a cluttered forest. This involves analyzing quite complex visual scenes, where familiar and unfamiliar objects combine and occlude other objects.

Rao et al. [36], which we will get back to later, identify two central problems the visual system must solve to correctly perceive the environment: object separation and segmentation. The first problem is identifying the high-level object categories in a scene consisting of superimposed objects, for example a cup on top of a table. So, the visual scene of “a cup on a table” needs to be recognized as being composed of two objects, “cup” and “table”, that have previously been presented to the system and learned, and not as an unknown object. This is the problem of object *separation*.

The second problem the visual system must handle is to identify the lower level features that correspond to the higher level objects they are part of. This would enable the handle of the cup to be identified as a feature belonging to the higher level category “cup”. This is the problem of *segmentation*. The biological significance of this ability is that it allows the identification of high-level percepts to be followed by an appropriate action, such as lifting the cup.

According to Rao et al., while separation of objects have been extensively studied in artificial neural network literature, the additional problem of segmentation has not. The reason being that feed-forward ANNs have an inherent problem with the segmentation problem, which we will get back to in the next section, and this problem has typically been attacked with non-neural approaches. Studying the brain’s solution to this could inspire new applications and advancement for instance in the field of robotics.

This is our primary motivation for this project.

In this thesis we develop a novel model that to a certain degree extends the traditional artificial neural network model with the capability of correctly separating and segmenting patterns, primarily in the problem domain of visual perception. The main idea has been that by extending ANNs rather than introducing entirely new paradigms, it could thus be possible to continue using effective tools developed for ANNs such as pattern classifiers and associative and auto-associative memory [38] but with increased power and flexibility.

To develop this model, our strategy has been to take primary inspiration from recent results in neuroscience and related computational models. Thus we will delve in to computational neuroscience in this thesis, and though we have tried our best to make it as accessible as possible, we apologize beforehand for any unknown concepts insufficiently explained. It took a great deal of time to arrive at our model, and this is reflected by the amount of background material and introductory experiments we have included in this thesis.

So, the primary research question in this thesis is *to study how the brain solve the separation and segmentation problem, and to construct a computational model able to do the same.*

Models able to do this has only recently appeared, and the usual approach is to employ oscillatory models which are quite computationally expensive. To the best of our knowledge no model has been able to do this in an efficient manner. We believe the model we develop in this thesis is a contribution in this direction, and has the potential to be extended to many other problem domains. As such, the focus of this thesis is the development of this model, and the experiments we perform on it are the very first initial explorations of our model's capabilities.

This thesis is structured as follows. This introductory chapter introduces the problem and hopefully imparts a high level of understanding of what we hope to achieve in this thesis. Chapter two, background, introduces some necessary concepts and existing models before we start developing our own model. The third chapter, methods and implementation, introduces our model in detail after a rather lengthy discussion of the components and background of the model itself. Chapter four describes experiments and results done on the model, while chapter five includes some more general discussions and observations. This document rounds off with conclusions and future work in chapter six. Additionally, we have added a small code example of our model in appendix A.

Chapter 2

Background

In this chapter we introduce some necessary concepts before we start developing our model. As our model is based on the biological mechanisms of the brain, we will approach various topics, from neuroscience to computational modelling, including even some philosophy.

2.1 The binding problem

Anne Treisman writes in [48] that “the world we effortlessly - and usually accurately - perceive consists of complex objects that are characterized by their shapes, colors, movements, and other properties. To identify an object, we must specify not only its parts and properties, but also how those parts and properties are combined”. We know that different parts of the human brain process different aspects of sensory and other information, so some mechanism is needed to 'bind' the information relating to each object together and to distinguish it from others.

The binding problem is a long-standing issue in the field of neural computation [41]. While it since has expanded to mean many things, the notion of a 'binding problem' was first used in the context of feature integration and perceptual segmentation [13]. Figure 2.1 shows the idea. As features are activated by some feature detection system, how does the rest of the brain know which features go together? This is the problem of feature integration. Perceptual segmentation on the other hand is the ability of the brain to distinguish separate objects from each other.

A related notion is Rosenblatt's superposition catastrophe. Frank Rosenblatt was one of the pioneers of the neural networks field with his development of the perceptron, a sort

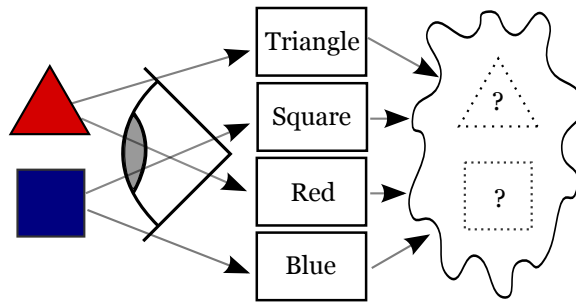


Figure 2.1: The binding problem. Visual inputs activate different feature detectors. However, the rest of the brain doesn't know which features go together.

of single layer classifier. In [39] he states that neural networks do not have the capacity to encode superimposed inputs. For instance, a network trained to distinguish between two patterns sharing common features would probably not be able to distinguish between these patterns if they later appeared simultaneously, as this would be outside the training distribution. This sharing of features is what causes the superposition catastrophe.

We demonstrate this by quickly sidestepping and explaining neural networks. We will refer to such networks in this thesis as an artificial neural network (ANN) to distinguish it from a biological neural network. An ANN consists of a set of units, called neurons, which are organized in layers and connected by links called synapses [8]. Usually there is an input layer, an output layer and some number of hidden layers. When used as a classifier, there is one output neuron per class. Figure 2.2 shows an example of an ANN with 4 input neurons, no hidden layers, and 2 output neurons. This particular network would be able to distinguish between 2 different input patterns, as it has two output neurons (we assume local coding for now).

How does classification occur? Some feature detector sets the firing rate of the input neurons to a specific value x . The weights of their links determine how much that particular neuron influences downstream (feed-forward) neurons. A neuron calculates its input and resulting output by the following equations:

$$inp_j = \sum_{i=0}^n x_i W_{ij} \quad (2.1)$$

$$x_j = \frac{1}{1 + e^{-inp_j}} \quad (2.2)$$

Equation 2.1 simply sums the product of upstream neurons and their link weights. The resulting output of a neuron is usually modified by an activation function as seen

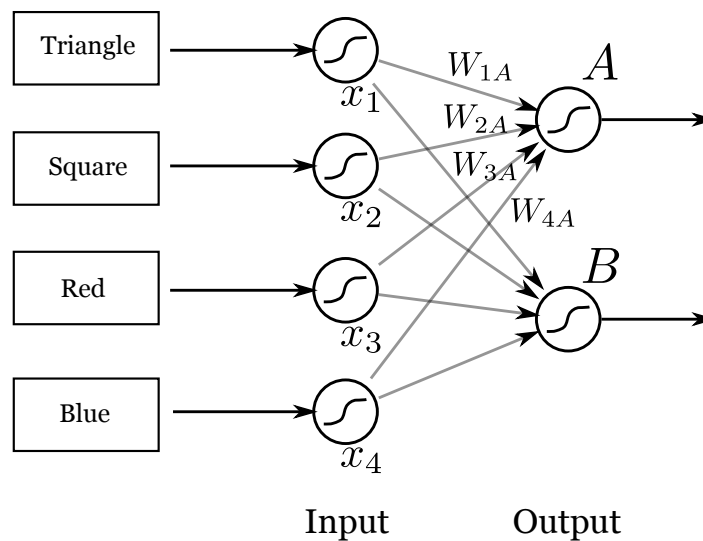


Figure 2.2: Artificial neural networks usually consists of an input layer, an output layer, and n hidden layers. Here, $n = 0$. Each unit has a firing rate x , and is connected to other units by a link with weight W .

in equation 2.2. This is the sigmoid function which is probably the most widely used activation function, which is why such units are regularly called *sigmoidal units*. Another name for an activation function is a threshold function, and is used to introduce non-linearity to the network. We will get back to this later, but suffice to say that such non-linearity results in many positive effects such as higher resistance to noise [38].

Finding the correct weights in an ANN can be a challenge. The common technique is to employ a supervised learning algorithm such as back propagation, which calculates the error at a neuron and modifies incoming weight to minimize that error. This can be a very time consuming process, and thus done offline in dedicated training periods.

Using figure 2.2, let us say that the weights are set up such that A fires when input neurons 1 and 3 are activated, and B fires when input neurons 2 and 4 are fired. Thus, A represents a red triangle and B a blue square. Now, imagine we present a blue triangle and a red square simultaneously to the network. The network, only having information about which features are activated, would respond with both output nodes on at the same time, representing a red triangle and a blue square. This is a binding error, so this superimposed input leads to the superposition catastrophe.

In this scenario we only have two possible categories. To add the possibility of classifying a blue triangle and a red square, we need to add two more output neurons. However, this leads to another problem which is the combinatorial explosion of output neurons

needed when number of input neurons increase. A cell which uniquely codes an input combination has been termed a “cardinal” cell, or more colorfully, a grandmother cell from the idea that a single cell fires when looking at ones grandmother.

One way of tackling these problems in an ANN would be to add a variable independent from the firing rate, which can provide additional information about the state of the units in the network. This is the idea behind the temporal binding hypothesis [51, 13]. Before we can understand that fully, we need to sidestep for a moment and introduce spiking neurons.

The sigmoidal units previously presented have a single variable which represents the *firing rate* of a neuron. The firing rate means the number of action potentials (AP) a neuron produces each time unit. An AP is a pulse of voltage which is created in the nerve cell, generally in the cell body called the soma, and propagated along its axon toward other nerve cells. We won’t go more in to the biology of nerve cells here, but an AP can be thought of as a single electrical impulse travelling from one neuron to another. Spiking neural networks (SNN) [35, 33, 24, 26] model each such impulse.

In the field of SNNs there are many different models to choose from. The original and still quite widely used model is the conductance based model of Hodgkin and Huxley [22]. It still survives as the most accurate model, but is very computationally intensive. The simplest to use is the family of Integrate-and-Fire neurons [45], but they suffer from unrealistic behavior. In this thesis, we use Izhikevich’s spiking neuron model [26] which was developed to strike a good balance between biological accuracy and computational performance. Izhikevich neurons are defined by the coupled differential equations:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I \qquad \frac{du}{dt} = a(bv - u) \qquad (2.3)$$

with after-spike resetting: if $v \geq \theta$ then $v \leftarrow c$ and $u \leftarrow u + d$

Here v represents the membrane potential and u represents a membrane recovery value. The parameters a , b , c and d define the type of neuron, see [26] for details on specific values for different types of neurons. This model can exhibit firing patterns of all known types of cortical neurons by choosing these 4 parameters carefully.

Figure 2.3a shows a single spiking neuron where the input starts at $t = 10$ and is constant $I = 20$. The red line is the membrane potential v and the blue line is the membrane recovery value u . Here $\theta = 100$, which means that when $v \geq \theta$ the model is reset, and a spike is recorded. Notice that even though the input is constant, the actual

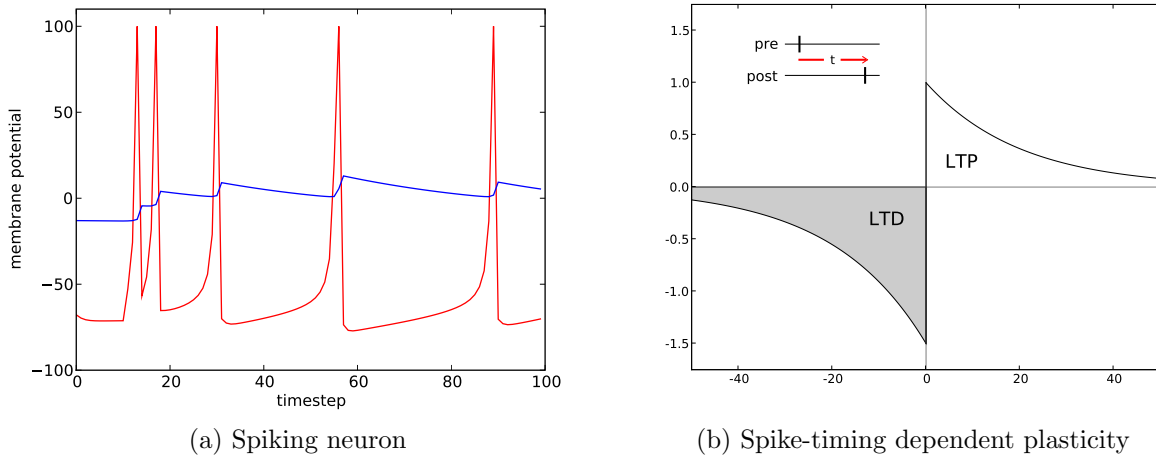


Figure 2.3: Spiking neurons and STDP. (a) Spiking neuron, Izhikevich style. Red line is the neurons membrane potential, blue line is the supporting membrane recovery variable in Izhikevich's model. Here, the neuron has a constant input. The refractory period (the time during which the neuron can not fire again) is regulated by the membrane recovery variable. This is what causes the increasing time between each spike. (b) STDP learning in a SNN. A synapse is strengthened (long-term potentiated, LTP) if an input pulse arrives before the neurons spikes. Otherwise, the synapse is weakened (long-term depression, LTD).

spikes the model produces vary in time.

Similarly to ANNs, learning in SNNs change synaptic weights. While there are multiple methods for synaptic plasticity, the most popular method seems to be STDP, or spike timing-dependent plasticity. STDP was developed formally by Song et al. in [43] and can be visualized as in figure 2.3b. The idea is that the magnitude and change of synaptic weights between a pre- and postsynaptic neuron depend on the timing of spikes: if the presynaptic spike arrives at the postsynaptic neuron before the postsynaptic neuron fires - for example, it causes the firing - the synapse is potentiated. Its weight is increased according to the positive part of the STDP curve but does not allow growth beyond a cutoff value. If the presynaptic spike arrives at the postsynaptic neuron after it fired, the synapse is depressed. Its weight is decreased according to the negative part of the STDP curve.

So, what is the benefit of this increased realism? Well, we have added another dimension to the neural network: *time*. The actual timing between spikes can be exploited to carry information. Indeed, SNNs appear to be able to give new computational solutions for temporal pattern recognition, as shown in experiments performed by Hopfield and Brody in speech recognition [24, 25]. However, the aspect we wish to focus on here is

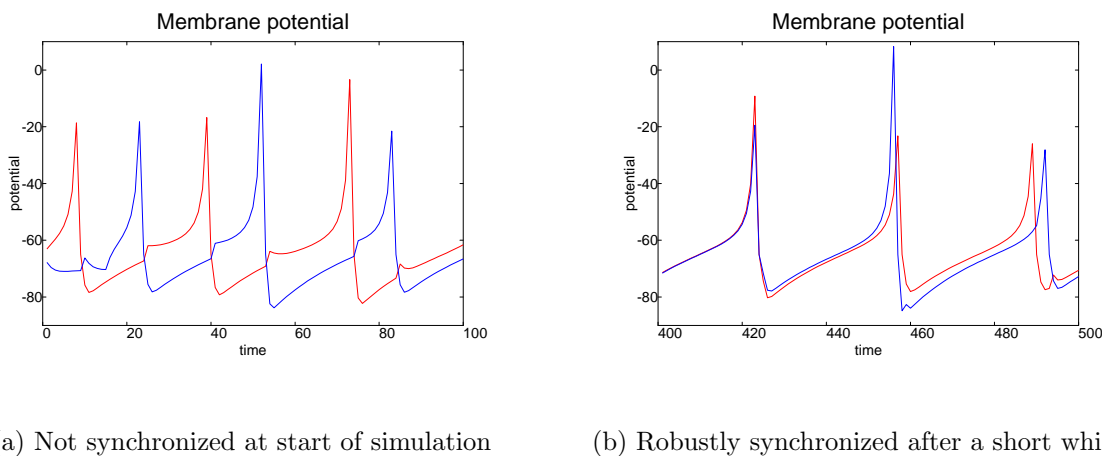


Figure 2.4: Synchronization of spiking neurons. Here, two neurons represented by red and blue line are weakly connected to each other. (a) The neurons start out unsynchronized. Notice the small bump on each neuron before they spike. That is the incoming spike from the other neuron; conductance delay is 1 ms. (b) Due to this input, they reliably synchronize after a short while. Notice that they do not synchronize quite exactly due to the conductance delay.

synchronization.

The neuron model in equation 2.3 can be thought of as a relaxation oscillator. A relaxation oscillator is a type of oscillator that gradually builds up some form of energy and at a certain point discharges all the energy, only to begin storing up energy again. So it can be compared with a capacitor. As Buzsáki mentions in [7], oscillators of this type synchronize robustly and with great stability. Figure 2.4 shows two Izhikevich neurons connected together that synchronize after a short while.

So, after this short introduction to SNNs, we can finally explain the temporal binding hypothesis. Von der Malsburg [51] proposed that the activity of low-order neurons would be combined only when their spike activity was synchronized to within a few milliseconds to create a synchronously active cell assembly [21, 1]. Synchronization would be dynamically modulated, so that a particular cell could belong to one cell assembly at one moment and to a second at another. Thus, the combinatorial explosion mentioned above could be avoided and a large number of categories could be encoded with a reasonable number of neurons. Later, Engel and Singer [13] extended this temporal correlation theory to their temporal binding hypothesis. They proposed that *populations* of cells that represent low-level features (for example, populations in the primary visual cortex) synchronize their activity when they respond to different elements that are to be linked in the analysis of

a scene.

In relation to our original problem of feature integration and segmentation, the temporal binding hypothesis proposes that the populations representing each feature synchronize their activity with the corresponding output population. Thus, the features 'red' and 'triangle' from figure 2.1 would synchronize their firing, while 'blue' and 'square' would also synchronize their firing. This would allow the brain to distinguish between the two cases, and the mechanism that keeps these apart is studied in the next section.

This temporal binding hypothesis has been criticized, most notably by Shadlen and Movshon [41], but remains the best explanation for how the brain solves the binding problem. One of the main criticisms of this theory is that while it describes the *signature* of binding, it does not detail how binding is computed. Since [41] however, multiple models have been proposed, and the model we develop in this thesis will also take issue with this.

2.2 Populations and oscillations

We believe that a lot of the critique of the temporal binding hypothesis stems from a confusion of processing levels. As Walter Freeman writes, there is just so much going on in the brain that one is easily astounded by the level of background activity. As such, it is easy to be sceptical about the notion of individual neurons being able to synchronize with others as they can receive inputs from over 10000 other neurons. The level of noise should just drown out any attempt of single neurons to synchronize with each other. However, neurons can form assemblies [21], synfire chains [1] or polychronous groups [28], and these populations of neurons can synchronize with each other even though individual neurons of those populations do not participate all the time. We will explore this perhaps counter-intuitive notion soon.

Here we define a population of neurons simply as a set of neurons able to respond collectively to some influence. This does not necessarily mean that the neurons in the population are neighbors anatomically, but that their connections and conductance delays are organized in a way so that they work as a collective unit. In our thesis, we will use such populations as the basic computational unit.

This notion of a population as the basic computational unit is not new. Vernon Mountcastle first proposed in 1978 [34] that the neocortex (the outer layer of the brain which evolved late in human evolution and the region that most separates humans from other

mammals due to the relative size) works on a common principle. This common principle was based on cortical columns (functionally defined regions of the cortex, with many thousands of neurons) as the basic units of computation. The exact function of these populations and resulting algorithm of the brain has been left unanswered. Jeff Hawkins, inventor of PalmPilot and hobbyist neuroscientist, was heavily inspired by Mountcastle and in his book 'On Intelligence' [20] proposed that the function of the brain was prediction. We won't promote this view in our thesis, however the book is highly recommended.

One of the most widely used models to model neural populations is the Wilson-Cowan cortical dynamics model [55]. The Wilson-Cowan model was developed to formulate the interactions between populations of excitatory and inhibitory neurons. These two types of neurons are different in that excitatory neurons increase a post-synaptic neuron's input level, while inhibitory neurons decrease it. In the Wilson-Cowan model, the activity of the populations are $E(t)$ and $I(t)$ for the excitatory and inhibitory populations, respectively. The dynamics of the model are roughly described with the following two equations:

$$\tau \frac{dE}{dt} = -E + (1 + r_e E) S_e [c_e E - g_e I + P(t)] \quad (2.4)$$

$$\tau \frac{dI}{dt} = -I + (1 + r_i E) S_i [c_i E - g_i I + Q(t)] \quad (2.5)$$

where r_e and r_i are the fraction of populations currently in their refractive period, S is the sigmoid function, c is the average number of excitatory synapses per cell and g is the average number of inhibitory synapses per cell. Finally, $P(t)$ and $Q(t)$ is the external excitation for E cells and I cells respectively. While relatively complex and with plenty of parameters, if these are tuned correctly several interesting phenomena can be exhibited. These are recounted to a large degree in Wilson's 'Spikes, Decisions and Actions' [54].

Another population model, while not in broad use but interesting nonetheless, is Freeman's K-set hierarchy [14]. It is both a model of neural population dynamics and a description of the architectures used by biological brains for various functional purposes. The lowest level of the hierarchy, the K0 set, provides a basic unit that models the dynamics of a local population of tens of thousands of neurons. A K0 unit models the dynamics of an isolated neural population. From the basic K0 unit architectures can be built up that capture the observed dynamics of increasingly larger functional brain areas. KI models excitatory-inhibitory populations roughly at the same level as Wilson-Cowan. KII models populations corresponding to organized brain regions, while KIII combine three or more KII populations to model functional brain areas such as the perceptual

cortex or hippocampus.

We do not use either of these models in our thesis, instead opting to develop a simple model of our own that better fits the dynamics we want to achieve. This begs the question however, why do we develop mathematical abstractions of populations of neurons at all? With increasing computational power, why can't we just simulate a large enough number of spiking neurons? Well, even if the exact organizational principles and dynamics of neurons were known, we are still nowhere near the required processing power needed to simulate a very large number of neurons for it to do anything interesting in real time. Currently, the two largest efforts are IBM's Blue Brain project and Izhikevich's project at The Neurosciences Institute. Both these projects have the goal of simulating a real brain, but Izhikevich [30] currently has the record of with 10^{11} neurons and 10^{15} synapses simultaneously simulated, roughly the size of the human cortex. However, it took 50 days of processing on a Beowulf cluster of 27 processors to simulate one second of model time. Given the, perhaps unlikely, continuation of Moore's law, this means we would be able to simulate a brain real-time some time around 2050. In the mean time it helps to use certain abstractions such as population models.

So, why do we even need neural populations as some sort of computational unit? Spiking neurons are capable of handling fine spatial and temporal scales as the integration period (the build-up period between spikes) is on the millisecond level. However, human behavior acts on information on much larger temporal and spatial scales [17]. To exemplify this distinction, a neuron that fires maybe 50 times a second has no memory of what happened 2 seconds ago, information that might be vital to the organisms behavior. So population codes are helpful to represent the larger temporal and spatial scales required to control adaptive behaviors.

Although a population consists of a set of neurons, populations can have very different dynamics than neurons alone. We will study this more in the next chapter, but the basic reason is that a population has, in addition to incoming and outgoing connections, a large number of internal recurrent connections with varying conductance delays. Also, Buzsáki points out [6] that a system consisting of excitatory and inhibitory neurons would tend to oscillate given the balanced struggle between these two types of neurons. The internal organization of the population, particularly the conductance delays (the time it takes for a spike to travel between two neurons), would determine the oscillatory frequency.

We saw earlier how single neurons could act as relaxation oscillators. The oscillatory behavior of populations however would work very differently, as harmonic oscillators. The system would be damped due to internal friction of synaptic weights, but would be driven

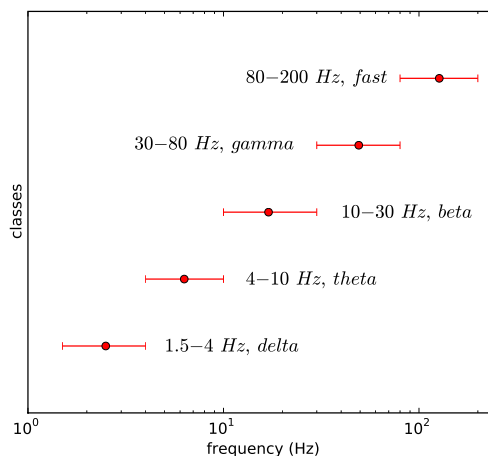


Figure 2.5: According to Buzsáki [6], neural networks in the mammalian brain demonstrate several oscillatory bands. The mean frequencies of the experimentally observed oscillator categories form a linear progression on a logarithmic scale. Data was experimentally recorded from a rat’s cortex.

by external connected populations. A harmonic oscillator is usually visualized as a spring: imagine a spring between your hands and vibrate your hands back and forth. If you time it just right, your hands are going to have a spring bouncing back and forth at a certain resonance. As Robert Desimone said in an interview in relation to a recent article in Science [19], “the neural equivalent of that is a very strong signal in the brain”.

This concept of resonant oscillatory activity is central to the dynamics of our model.

Buzsáki is a huge proponent of oscillatory synchronization, and he goes so far as to say that “the synchronous activity of oscillating networks is now viewed as the critical middle ground linking single-neuron activity to behavior” [7]. Indeed, he calls this an emerging new interdisciplinary field (“neuronal oscillations”) that cuts across psychophysics, cognitive psychology, neuroscience, biophysics, computational modeling, physics, mathematics and philosophy. One of his central theories is that the brain consists of a system of oscillators, working within different oscillatory bands, see figure 2.5. The mean frequencies form a linear progression on a logarithmic scale with a constant ratio between neighboring frequencies, leading to separation of frequency bands. Neighboring frequencies are typically associated with different brain states and compete with each other, but several rhythms can also temporally coexist and interact. The overarching point is that these oscillatory patterns enable the brain to operate at multiple temporal and spatial scales simultaneously.

Let us briefly clarify what we mean by the linking of single-neuron activity to behavior.

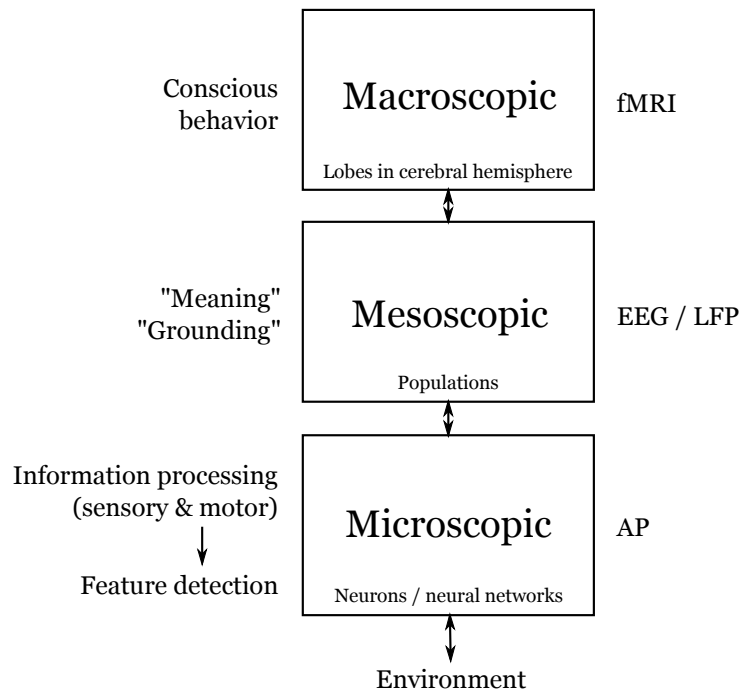


Figure 2.6: Levels of processing in the brain as presented by Freeman's theory of meaning in perception. The boxes represent levels of processing, with the column to the left of the boxes denoting corresponding type of processing done at that level. The column to the right of the boxes denote the typical tool used to measure activity at that level.

To do that we employ Freeman's theory of meaning in perception [15] shown in figure 2.6. The boxes represent levels of processing in the brain. The macroscopic level represents the behavior of the organism and entire lobes in the cerebral hemisphere are involved at this level of processing. This level can be thought of as the domain of symbolic processing, and thus symbolic AI. The microscopic level on the other hand is where interaction with the environment happens, and where the information from the environment is detected using neural networks as feature detectors. This level is about sub-symbolic processing, and thus sub-symbolic AI.

However, the information at the microscopic level needs to be made available to the behavioral processes in the macroscopic level somehow. Freeman has some ideas of how this is done of course, and points out that *meaning*, not information, is the currency of the mind. Further, he states that the information processing metaphor of the brain has dominated neurocognitive science for half a century and it is time to think differently. So, the information has to be translated to meaning somehow, and meaning needs to be represented differently than information. He compares this translation to a fundamental state change, such as when steam condenses to water. Indeed, he introduces something he

calls a wave packet [16] which is the equivalent of neural activity after this state change. Presumably this is the mixing of environmental information with internal knowledge. This would then represent the mesoscopic level.

Whatever one thinks of this theory, it is a useful framework to represent different levels of processing in the brain. Many other authors write about the symbolic and sub-symbolic level being bridged by a third one. For instance, Gärdenfors [18] bridges this gap with a layer of geometrical representations of the information, so the higher levels can distinguish between degrees of features.

This philosophical sidestep was useful to put in context that the brain presumably works at different levels. So Buzsáki's claim of synchronous oscillations linking single-neuron activity to behavior could be compared to a state change where the higher level processes of the brain communicate by the synchronized oscillatory activity of populations of neurons rather than single action potentials. Of course, the action potential is still what enables the communication, but the aggregated activity of populations is invisible to each single neuron.

To sum up this section, populations of neurons have distinctly different activity than single neurons. They can synchronize their oscillatory activity with other populations and the resulting resonance is a strong signal which overcomes noise in the brain. In relation to our original problem in figure 2.1, we can now see that population representing 'red' and 'triangle' would oscillate at one frequency, while 'blue' and 'square' would oscillate at a different frequency. This is what binds the representations together and separates them from other representations active at the same time. These ideas are what form the primary inspiration of our model.

2.3 The importance of feedback

We still haven't touched upon the problem on what exactly causes the populations to synchronize their oscillations. The popular view is that higher processing regions, having more information of the context through bigger and more complex receptive fields, somehow influence lower regions. As this is integral to our model, we visit this notion briefly here.

Normally, classification networks such as ANNs use feed-forward processing. This means that the processing proceeds through a series of layers, input patterns entering at one end and output patterns leaving at the other, thus forming a mapping from one pattern

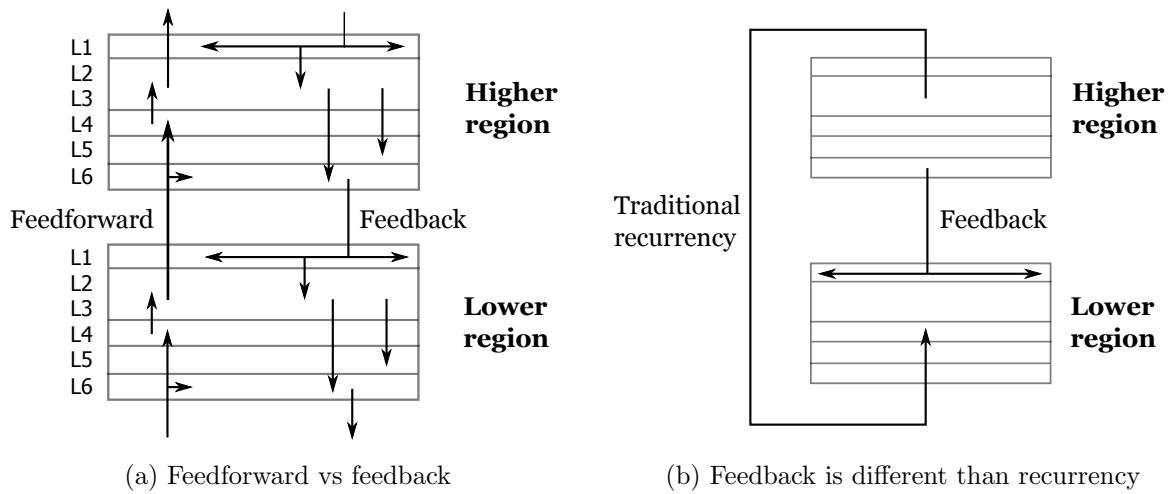


Figure 2.7: Simplified cortical connections. The cortex has a laminar structure consisting of 6 layers. (a) Feed-forward connections are different than feedback. Feed-forward follows a straightforward path where connections from lower regions enter at layer 4 and continue to layer 2/3 before going up to the next region. Feedback is less direct, but the main feature is in layer 1. The feedback from higher regions have a large degree of fan out in layer 1 and can cover large distances. (b) Seeing that feed-forward and feedback is handled quite differently, we make a distinction between recurrency as used in traditional recurrent neural networks and feedback.

to another. However, when studying anatomical connections such as in the primary visual cortex, there is a very large degree of feedback connections. Some authors put the number of feedback connections as high as 80% of the total connections. Clearly, for the brain to devote so much space for these connections, they must perform some important function. See figure 2.7 for a simple schematic of the difference between feed-forward and feedback connections in the cortex.

There is some debate on the role of these connections. Hawkins [20] believe their primary function is predictive, in that a cortical region receives both actual input from the environment (feed-forward) and predictions from earlier experiences (feedback) and compares them. He subsequently created a computational model called Hierarchical Temporal Memory (HTM) that utilizes Bayesian networks to perform machine learning. He also started a company called Numenta to capitalize on this technology.

Grossberg in his Adaptive Resonance Theory (ART) [10] uses feedback connections in pattern matching, not as predictions, but as expectations. The difference is that prediction says something about the future, while expectation in this sense is the matching of an activated representation with its current input pattern. His system has two layers, F_1 and F_2 , where input arrives at F_1 and is classified at F_2 . ART has two sets of weights,

one feed-forward and one feedback. F_2 has a competitive aspect such that only a single category can win. Additionally, a vigilance parameter controls how patterns are matched.

The dynamics of ART are essential to how it functions. When F_2 categorizes an input, it sends feedback to F_1 what the exact stored pattern is, the expectation. If there is a match in F_1 , then the routine completes. If not, then the vigilance parameter is disinhibited which causes another iteration where another pattern is chosen in F_2 . This search goes on until a pattern is found that matches (in the case of noisy patterns), or the input pattern is learned in which case a new category is created in F_2 (in the case of novel patterns). This dynamic solves what Grossberg calls the “stability-plasticity dilemma” which concerns how brains can quickly learn to categorize information about the world, and to remember it without experiencing catastrophic forgetting.

Both Hawkins and Grossberg use top-down influence as essential ingredients of their systems. However, Shadlen and Movshon point out that this kind of feedback “would require exquisitely precise and dynamically configurable connections” [41], and that “available data on feedback connections is that their organization is far too crude for this purpose”. Indeed, as shown in figure 2.7a, the feedback connections in layer 1 have such a large degree of fan out that targeting individual neurons seems very difficult. Francis Crick, co-discoverer of the DNA molecule, suggested instead that these feedback projections are designed to modulate neuronal activity in larger regions [12], something we interpret as the population level.

The common view in neuroscience now is that feed-forward connections mainly drives activity, while feedback connections are mainly modulatory. The effect is that higher regions having access to a bigger picture of the input can modify and shape activity earlier in the processing to account for context. This is a large departure from earlier computational models where feedback usually equaled recurrency (see figure 2.7b).

In a very recent paper that has already gained a lot of interest, Gregoriou et al. [19] used paired recordings in subsequent regions of a monkey brain to discover that stimuli in their joint receptive fields leads to enhanced oscillatory coupling between the two areas. This means that they found evidence that a stimulus can indeed synchronize two areas of the brain, even over large distances. What was even more impressive, was that the coupling appeared to be initiated in the higher region. Their analysis suggest that top-down inputs dominate at the onset of the stimuli, but bottom-up inputs come to predominate over the course of sustained activity. The oscillations that coupled these two regions was thus initiated by higher firing rates in the higher region.

This does give strong experimental evidence that the higher regions indeed influence

lower regions by modulating their firing to synchronize oscillations with other regions. We are highly influenced by this result, and our model incorporates these ideas.

2.4 Existing computational models

So, we've seen how the binding problem, particularly perceptual segmentation and feature integration, can be solved by oscillatory synchronization. We also saw in the previous section how higher regions can influence lower regions. The obvious implication we are pointing at is that top-down connections modulate lower regions in a manner to synchronize their oscillations. So far we have only seen the signature of how this synchronization occurs. In this section we review two computational models that exploit top-down processing to achieve separation and segmentation of patterns, but do so in very different ways.

Our first stop will be Rao et al.'s dynamical units [36]. In this paper they review many other approaches since von der Malsburg's [52] influential paper which formulated the theory of using synchrony as a solution to segmentation. Without reproducing that review here, the general theme is that most approaches based on von der Malsburg (for instance [11] and [53]) inherit some of the shortcomings such as reduced biological plausibility due to global inhibitory units and the inability to disambiguate objects with partial overlap. Rao et al. position themselves as the pinnacle of these models using oscillatory synchrony and are thus representative of most of them. However, they include an interesting observation by Lee [32]. He observes that most of the existing models for oscillatory neural networks are "either too simplified to simulate any real chaotic neural behaviors or too complicated to be applied as feasible artificial neural networks for applications". Rao et al. continue that this means there is an issue with modelling at the right level of abstraction. What they mean here is similar to what we saw in section 2.2; one needs to apply the correct techniques for the level of computation one is at. Indeed, they mention that there are few attempts in literature that have succeeded at this goal. They explicitly try to address this in their paper, but we will come back to why we think they also fall a bit short.

Basically their system consists of a set of units that oscillate at certain frequencies, and through interactions with other units are able to synchronize their activity. The units are organized in two layers: one input layer and one output layer. The input layer receives input, and the output layer is trained to respond with a winning category unit for the input. The input layer feeds activity to the output layer in a feed-forward

fashion, but the output layer also feeds activity back down to influence the input layer. In addition, the output layer has competitive lateral connections which is controlled by a rather complicated objective function to ensure sparse coding. Sparse coding is simply a form of distributed coding where a small number of units represent some input pattern (which is different than the cardinal cell local coding we saw earlier).

This model is trained using an extension of the traditional Hebbian learning rule. The model has a training period and a testing period. During the training period, the weights are kept fixed and the time-scale is slowed down, and a sort of gradient descent over the synaptic weights is performed only after the network settles.

Through the use of feedback and synchronization of oscillators, the system is able to segment inputs to their resulting categories. This dynamic works by the units having the largest amplitude in the output layer also has the largest influence on synchronization of the units in the input layer. So, after the network settles, input units are phase synchronized with the corresponding output unit they most contribute to. They train their network on single patterns, and test their network on pairs of input patterns. They go on to show that their network is capable of successfully segmenting input patterns with approximately 80% success rate. Their networks are also able to separate patterns, in that the number of highly activated output units represent the number of simultaneous patterns. The input patterns they experiment on are simple visual figures on an 8 by 8 pixel grid.

We are highly influenced by this paper. However, we were never able to get a working implementation of this system due to the complicated objective function required. Our system has a similar dynamic with using feedback from output nodes to influence input nodes, which is the absolutely essential mechanism to correctly segment input patterns. We have not presented our system yet, but we were able to get better results without the use of any complicated objective function while still being able to separate and segment correctly. One big difference between our systems is that we primarily use supervised learning while Rao et al.'s system uses unsupervised learning. We will get back to the difference between these types, but this is not the entire reason why we got the better results.

As mentioned, we think their system also misses the correct level of abstraction to capture the temporal dynamics of the neurons. With reference to figure 2.6, the mesoscopic level basically revolves around populations of neurons. Rao et al. essentially equate a population of neurons as a single oscillator, and in addition has these single oscillators work directly as feature detectors. While the latter could just be for demonstration pur-

poses, the former might be a bigger problem. In our system we equate a population of neurons with a population of oscillators. We will see later what advantages this gives us but allows more rapid and robust synchronization. Indeed, while Rao et al.'s system could spend hundreds of time steps to settle, our network typically settles in 10-15 time steps.

Rao et al.'s solution is representative of oscillatory solutions to the segmentation problem. In a follow-up paper, they create a three layer system which they use to compute optimal sizes of feed-forward, feedback and lateral connections [37] which is rather impressive but creates an increasingly complex equation set as each layer needs its own equations. All in all however, this model is a major influence to ours.

Another model that is heavily influenced by feedback processing is Achler's elegant regulatory feedback system [2]. This system was primarily motivated by what Achler calls "overwhelming evidence for feedback inhibition": neurons that can inhibit their own input. Through this idea he is able to construct a model able to disambiguate simultaneous patterns using the dynamics of his model. A interesting point he makes is that the method of feedback he uses results in "feedback homeostasis", which means the system is regulated by some equilibrium rather than relying on parameter optimization. Indeed, he cuts his model down to the bare bones: there are no parameters, not even synaptic weights: it is clear he thinks weights are part of the problem with connectionist models. Whatever one thinks of his assumptions, he does present a very interesting model.

His system consists of two layers, one input layer and one output layer without any hidden layers. All connections in his system have an implicit binary weight, 1 or 0, where 0 really means it is absent in the model. The input layer drives the output layer, and the output layer inhibits this driving activity. While that seems counter-productive, it allows for some interesting dynamics. His system is governed by three equations:

$$y_a(t+1) = \frac{y_a(t)}{n_a} \sum_{i \in N_a} f_i \quad (2.6)$$

$$f_b = \frac{x_b}{Y_b} \quad (2.7)$$

$$Y_b = \sum_{j \in M_b} y_j(t) \quad (2.8)$$

In these equations, x represents activity of input nodes and y activity of output nodes.

Equation 2.6 is the nonlinear difference equation governing the activation of output node a . Equation 2.7 is the pre-synaptic inhibition function affecting every input node b , and equation 2.8 is the feedback term. Notice there are no parameters and no weights. N_a and M_b denote all nodes connected to node a and b respectively, and weights are assumed as 1. Achler means this simplifies connectivity and training.

His goal is similar to Rao et al. The system is trained on single instances of input patterns and are tested on combinations of multiple simultaneous patterns. While this model does not segment patterns in the fashion as we have defined, it is able to separate patterns very effectively. He seemingly does not limit the type of input patterns presented to the network.

These equations define a sort of negative feedback relationship commonly studied in engineering control theory. In short, the model works by cycling activation between inputs and outputs. Inputs activate contending output representations which in turn inhibit their representative inputs. Inputs utilized by multiple output representations are more ambiguous and are inherently inhibited more. The inhibited inputs then affect output representation activity, which again affects inputs. The cycling is repeated until a steady state is achieved.

The training of this network is slightly unorthodox. Using a supervised learning principle, an input pattern is presented to the network and an output node is selected. Connections are formed between input node and output node if the input node is currently activated. If not, no connection is formed. This basically results in a set of binary weights. This type of learning is not as generalizable as conventional algorithms, and Achler admits that the model's inherent nonlinearity leads to the difficulty of forming general learning rules. He has not yet presented a better learning rule.

Here we can see how this system of top-down influence differs from traditional feed-forward categorizers such as ANNs. Where the ANN would effectively map an input pattern to an output pattern in a feed-forward fashion, this system of equations settle on a solution after a number of iterations. The dynamics of the model are such that feedback modifies input values which are subsequently modified by further processing. As Achler puts it, "the iterative nature allows robust inference during the recognition phase".

He points out that previous solutions for separating patterns and addressing the binding problem and superposition catastrophe has used oscillatory dynamics. Through the dynamics of his model, he is able to do similarly without the use of any oscillator. He goes on to show how his system is able to correctly classify simultaneous patterns when only trained on single patterns.

This model is impressive in its simplicity, but does work at a lower level than Rao et al.'s model. Its non-usage of oscillatory units does mean that while it is able to separate patterns, it is not able to segment them. Thus a part of the binding problem still persists. However, we include this model here as it shows clearly that the dynamics of top-down influence is able to solve our original problem. Also, by not using oscillatory units at all he is able to achieve a very simple system which is very computationally efficient. We are highly inspired by the elegant simplicity of Achler's system. While it does have its shortcomings, it did inspire us in what it was capable of without the use of oscillatory units.

The models of Rao and Achler form the main inspiration to our model. Our model attempts to build on the simplicity of Achler's model, particularly its dynamics, while retaining the flexibility of Rao's model. We start building the model in the next chapter.

Chapter 3

Methods and implementation

This chapter is more or less divided in two parts. The first describes the development of our model, including some exploratory experiments. This part is not a series of rigorous proofs which result in a mathematical model, but more a collection of observations building on the previous chapter. The second part describes the model in detail, and the manner of experiments we performed on it.

3.1 Populations of spiking neurons

We have previously defined a neural population as set of neurons connected to each other forming some sort of functional unit. Some characteristics include high degree of internal recurrence, inclusion of both excitatory and inhibitory neurons, varying conductance delays, and even random network topology. Without knowing how this subnetwork is connected, it can seem difficult to come to any conclusion of what this functional unit does.

However, in this chapter we will start looking at populations as some sort of abstract entity that does perform as some unit, perhaps even without the same physical neurons participating every time. This is a natural consequence of oscillatory connections between regions of the brain, and here we will start studying the oscillatory properties of neural populations.

The basis for all our work with spiking neurons has been the work of Izhikevich. Izhikevich has repeatedly refined his network of spiking neurons, from networks without plasticity [27] and networks with STDP [28] to networks with a more active form of reinforcement learning [29]. With the exception of the most recent one, his motive has

been to study macro level effects found in the brain such as gamma rhythms, while remaining biologically plausible. One discerning factor in these networks is the specified conductance delays between neurons, which he argues is key to the network dynamics he experiences. He mentions that careful measurements of axonal conduction delays in the mammalian neocortex showed that they could be as small as 0.1ms and as large as 44ms, depending on the type and location of neurons. Further, the propagation delay between any individual pair of neurons is precise and reproducible with a sub millisecond precision. As Izhikevich argues, why would the brain maintain different delays with such precision if spike timing was not important? In his simulations, all excitatory synapses are set to have a conductance delay between 1ms and 20ms and all inhibitory synapses have a fast 1ms delay. His simulations are discrete-time with a time resolution of 1ms.

Being a computational neuroscientist, Izhikevich discusses all his results in the context of mammalian brains. However, even though his networks achieve realistic results, his system is kept simple, easy to understand and implement, while being reasonably efficient. This is why we chose his system to simulate neural populations. In the following, we set up networks consisting of 1000 Izhikevich neurons, where 80% are excitatory and 20% are inhibitory, representing a typical cortical minicolumn. The networks are defined as sparse, meaning that all neurons are connected randomly with 10% probability so there are 100 synapses per neuron. This network with 1000 neurons and 100,000 synapses represents the layer 2/3 part of a cortical minicolumn.

Figure 3.1 demonstrates the Izhikevich system as described in [28]. From the initial system activity as seen in 3.1a, the system evolves toward a state with intermittent gamma frequency oscillatory activity. The reason of course being the STDP learning rule which tunes the weights according to network activity. The initial conditions for synaptic weights are set at $+4mV$ for excitatory neurons. During the evolution of the system, these weights are distributed toward $0mV$ and $+10mV$ which is exactly as expected, as [31] describes that synaptic weights in such systems are expected to distribute toward 0 or the maximally allowed value after some time of network operation.

The interesting point to notice here is in figure 3.1c. Even though we have oscillations in the gamma frequency range (20-100 Hz), each excitatory neuron only fires approximately 4 Hz in average (inhibitory neurons spike more often, around 16 Hz). This means that this population of neurons is able to oscillate at certain frequencies *even though not all neurons are involved in all phases!* This is a very interesting observation, which enables populations of neurons to have different dynamics than the set of neurons they consist of. Populations can thus transcend the limited time integration periods of single neurons. Of course, the oscillations of populations are not at all independent of the neurons, but

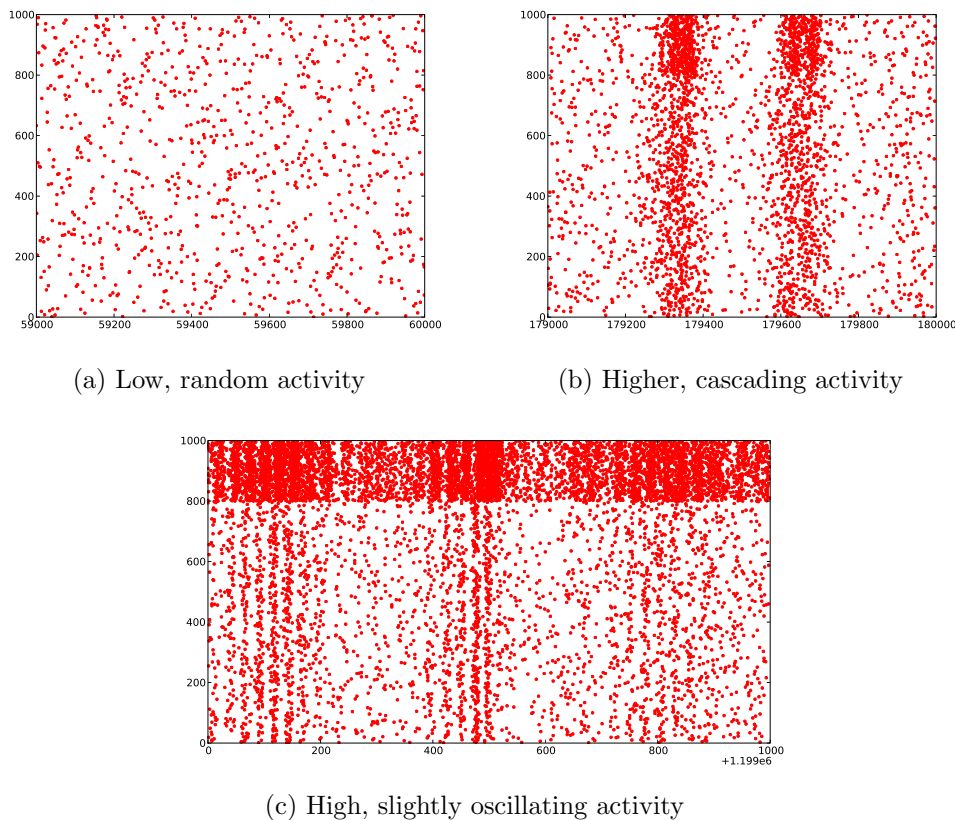


Figure 3.1: Raster plot of an Izhikevich network at various stages in STDP training. (a) At the start of training, there is mostly random activity, with low average firing. (b) After 180 seconds, a clear strengthening of synapses is evident as activity is higher and at certain periods crosses a sort of threshold which recruits most of the neurons to fire. These can be identified as delta oscillations, firing in the 2-4 Hz frequency range. (c) After 20 minutes model time training, oscillations in the gamma range (20-100 Hz frequency range) become evident, but are short lasting.

one can think of this as the population recruiting neurons to the oscillation as necessary. If not sustained by some external force, the oscillation quickly dies out due to intrinsic noise and the friction caused by thresholds.

Taking this abstraction further, one can view a population as a large set of possible oscillators working in parallel. These oscillators would continuously synchronize and desynchronize with each other, thus for example creating these transient gamma frequency oscillations. We will come back later to the question of synchronization, but for now we will hold on the image of a population of neurons as a system of oscillators. In a population of 1000 neurons, any set of neurons can transiently form an oscillator with any other set of neurons, giving rise to an almost infinite amount of possible oscillators. This complex relationship between sets of oscillators makes such populations, and thus the

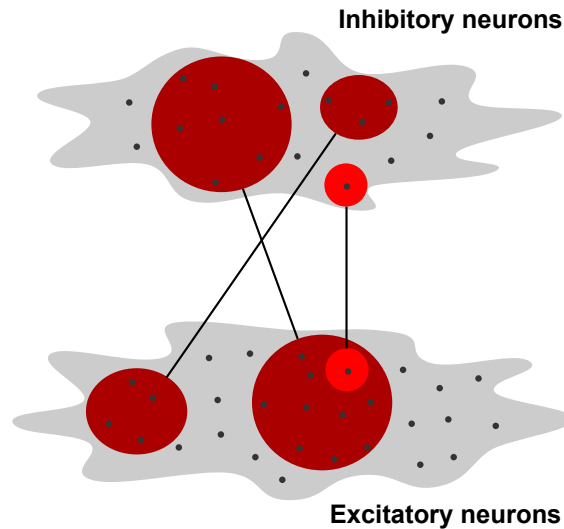


Figure 3.2: A neural population can be viewed as a system of many oscillators. Neurons can transiently form oscillators with other neurons, thus giving rise to an enormous amount of possible oscillators.

brain, a very noisy place.

So, if we take this view, without knowing the connectivity of a population and the type of neurons it consists of, how can we even begin to analyze the activity going on in such populations?

3.2 Analyzing population activity

The long standing method of analyzing activity in a group of neurons is using signal analysis methods on the local field potential (LFP) of a group of neurons. The LFP of a group of neurons is the extracellular potential caused by synaptic transmitter substances, and is thus not directly a measure for spiking rate, but rather the activity at the synapses. This is of close relation with the firing rate as a high LFP means that the group of neurons have a high input thus firing more often. In human studies, the LFP is usually measured by small electrodes attached to the scalp. This method is called electroencephalography (EEG) and is one of the oldest methods of measuring brain activity. The EEG was used to find the first evidence of brain waves. In Berger's 1929 paper [5] which established the use of EEG, he also gave name to the alpha rhythm and the faster and smaller amplitude beta rhythm.

In Izhikevich's system, the LFP is simply the sum of all inputs to all neurons. By

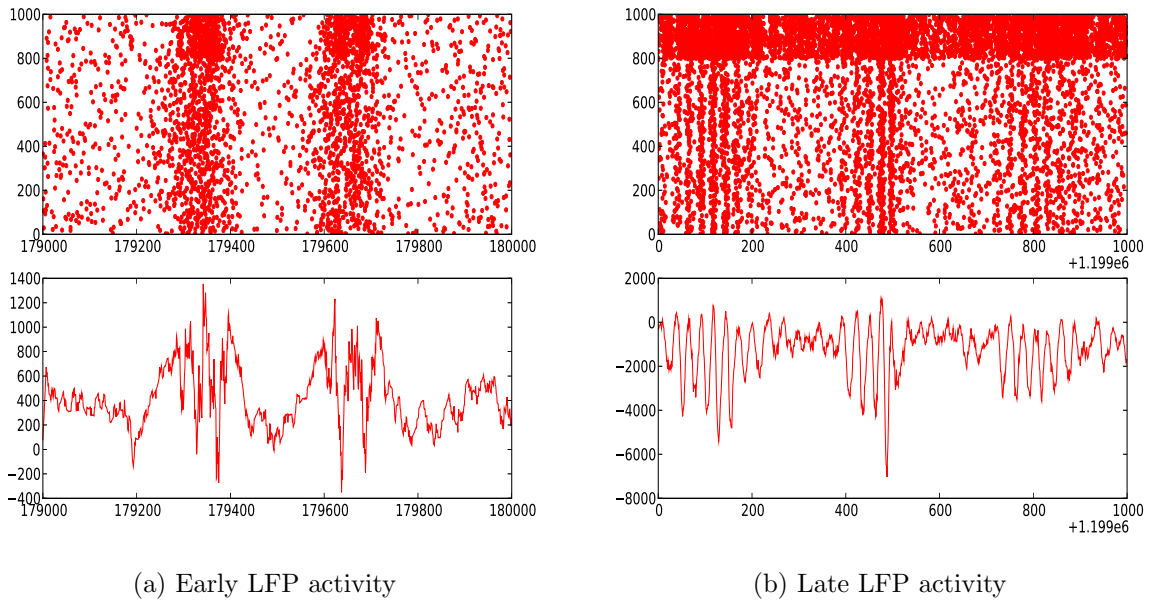


Figure 3.3: Local field potential (LFP) measurements of network activity at same stages of training as in figure 3.1. What is immediately visible is that the network seems to contain oscillations of many different amplitudes and frequencies.

using this measure, we can transform the n -dimensional system of firing neurons to a 1-dimensional signal. This signal can then be analyzed using signal analysis such as the Fourier transform. In figure 3.3, we have plotted the LFP alongside the raster plot firing diagram from figure 3.1. From this figure, one can see that the activity constantly fluctuates up and down, and that there seems to be oscillations of many different amplitudes and frequencies. To analyze this, we find the discrete Fourier transform (DFT) of the LFP signal.

The discrete Fourier transform is defined as:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad k = 0, \dots, N - 1 \quad (3.1)$$

While a basic and well known instrument, it helps to scrutinize the elements in this function to see why it is helpful in our analysis here. X_k is the power of frequency k . N is the number of samples we have, for example over a 1000ms time period we have $N = 1000$ as our time step is 1ms. The measured amplitude of the signal at time step n is x_n , and finally $e^{-i2\pi k \frac{n}{N}}$ is the comparison to what frequency k should be at time step n . This is more evident given Euler's formula:

$$e^{i\varphi} = \cos \varphi + i \sin \varphi \quad (3.2)$$

So, the DFT gives us the *power* of a signal at given frequencies, allowing us to analyze the different frequency components of a signal. Whenever one uses a Fourier transform, some sort of aliasing usually occurs. As we have a sampling rate of 1000Hz, the Nyquist-Shannon sampling theorem [42] states that we can not represent frequencies greater than 500Hz. But this is of no concern to us, as the rhythms we are interested in are of much lower frequency. But we need a method of handling aliasing at the lower part of the spectrum. The method to get a reliable reading of the power spectrum is to take a signal of a certain length, divide it into a number of blocks, use a windowing function to decrease this aliasing (also called spectral leakage), take the DFT of the block and average this across all blocks. We use the Hann windowing function, which is the most widely used windowing function. To get the *power spectrum*, we modify the DFT somewhat:

$$X_k = \left(\frac{1}{\sqrt{2\pi}} \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \right)^2 \quad k = 0, \dots, N-1 \quad (3.3)$$

We take a 10000ms slice of LFP activity and use blocks of size 1000ms, thus averaging across 10 blocks to get a reliable reading of the power spectrum. Figure 3.4 shows the network activity over this period. In particular, figure 3.4b shows a prominent peak in the 30-40 Hz region, which is defined as gamma rhythms. As can be seen in both power spectrum graphs, the trend is declining in the log-log graph, in a manner close to a straight line. This is defined as pink noise in signal analysis, and is characterized by so-called $\frac{1}{f}$ behavior, due to the straight line it forms in a log-log plot. Buzsáki argues in [6] that spectrums that follow a $\frac{1}{f}$ behavior (also called a power law) usually means some sort of self-organization. We won't analyse this further here, but this idea comes from the seminal work of Per Bak and his self-organized criticality [4].

There are two main observation we would like to make here. The first is that all oscillations are transient. They come together fleetingly and then disappear, only to reappear later. The system is thus very noisy, and the saliency of oscillations is necessary to overcome this noisy environment. The second observation is that all frequencies are represented in various degrees in a population, even though some are more prominent than others. One cannot treat this population as a single oscillator that oscillates at an exact frequency.

The reason we point out these two rather obvious observations, is that most models of oscillation in the brain use some form of oscillator that has a certain given frequency,

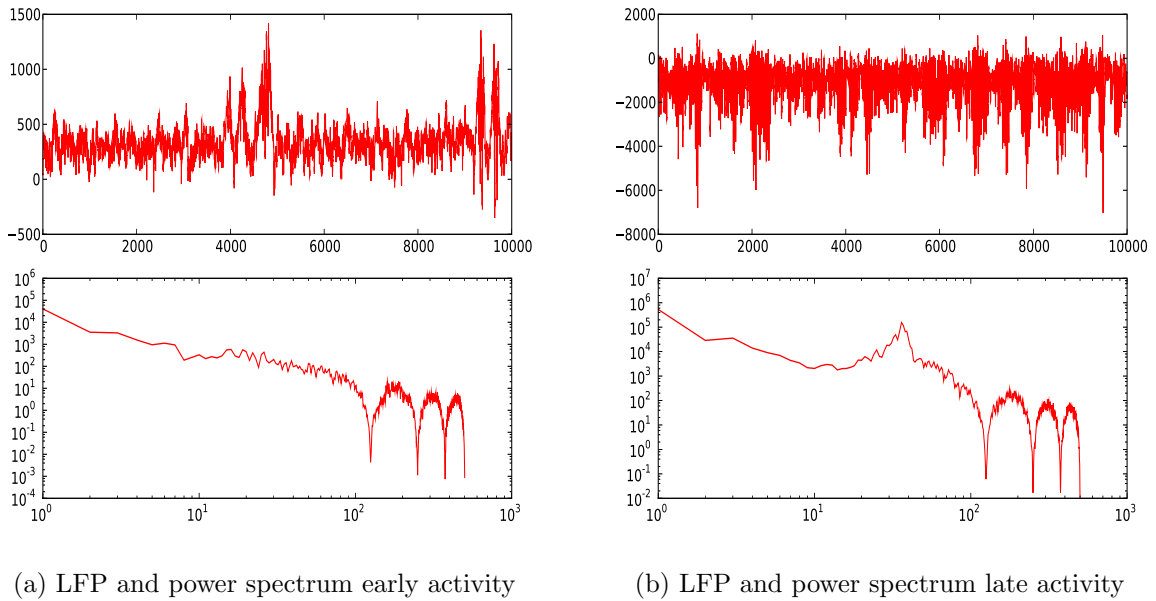


Figure 3.4: Local field potential (LFP) measurements and resulting power spectrum of network activity at same stages of training as in figure 3.3. Here it becomes evident that the population indeed has multiple oscillators at many different frequencies. At the later stage in (b), a prominent peak can be seen in the power spectrum around 30-40 Hz. This is gamma frequency. The power spectrum graph is log-log.

notably Rao et al.’s dynamical units we presented earlier. This is a very idealized situation. In the following, we use the fact that populations can oscillate in different frequencies as a feature.

3.3 Populations as systems of oscillators

At this point, we are looking at a population of neurons as a system of oscillators. Let us abstract that a bit, and focus on such a system. Here we define an oscillator generally, as follows:

$$\theta(t+1) = \omega + \theta(t) \quad \text{and} \quad x(t) = A \cos \theta(t) \quad (3.4)$$

Here, $\theta(t)$ is the phase of the oscillator at a given point in time, and ω is the frequency. Figure 3.5 gives a graphical depiction of this, and the intuition is that ω is added to the current phase at every time step. The amplitude of the oscillator is given by $x(t)$, and A is the maximum amplitude. In our examples here we set $A = 1$ for simplicity. We

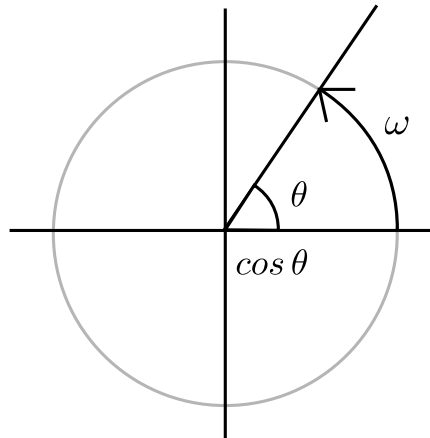


Figure 3.5: A graphical depiction of the generalized oscillator given in equation 3.4.

purposefully neglect connections between the oscillators for now, but this generalized expression for an oscillator will aid us later when we talk about synchrony.

A system of these oscillators would be a set with varying ω . In accordance with the system of Izhikevich neurons, we simply define that the signal we analyze is the sum of the amplitudes of all oscillators at any time step:

$$X(t) = \sum_{n=1}^N \cos \theta_n(t) \quad (3.5)$$

We treat this 1-dimensional signal in the same manner as in the previous section, and analyze it with the discrete Fourier transform and the resulting power spectrum. We create a system of 1000 oscillators, each with a ω randomized as to create frequencies from 0 to 100 Hz. To create the same sort of $\frac{1}{f}$ behavior as in the previous section, we bias this randomization toward 0 Hz, meaning we have more oscillators at low frequencies than at higher frequencies. This is done by simply squaring the random ω . We then randomize the starting phases, and sum their amplitudes for each time step according to equation 3.5. We observe this behavior for 10000 time steps as before, and calculate the DFT and power spectrum over 10 blocks of 1000 time steps and average their values. The results are depicted in figure 3.6.

Initially, as in figure 3.6a, we have a uniform set of 1000 oscillators biased toward oscillation at lower frequencies. This is clear in the log-log power spectrum plot as the power declines with higher frequency, confirming that we have more oscillators at lower frequencies. Then we add a signal with a Gaussian bias around 30 Hz and another signal with similar bias around 60 Hz, which is clearly visible as the peaks in figure 3.6b. An

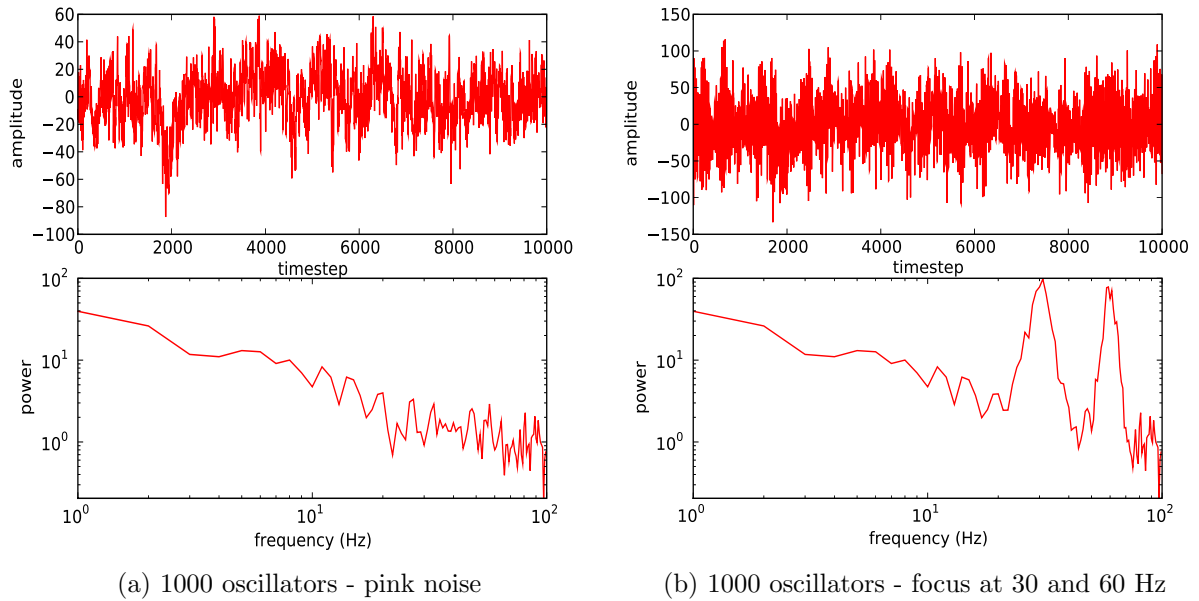
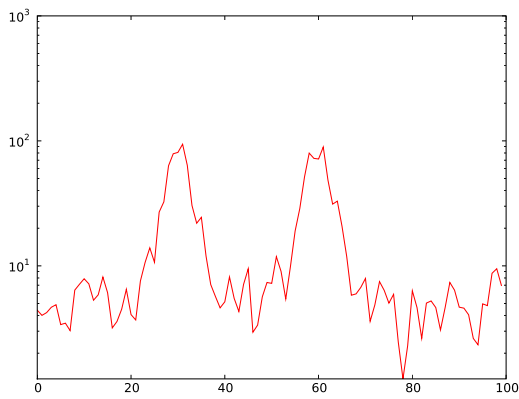


Figure 3.6: Power spectrums of systems of oscillators. Top: sum of amplitudes of all oscillators at each time step. Bottom: power spectrum of signal. (a) A uniform system of random oscillators. In this system, ω has been set to produce oscillators with frequencies from 0 to 100Hz with a bias toward 0 Hz to create a sort of pink noise. (b) Here, bias is toward a region around 30 Hz and 60 Hz, which can clearly be seen as the peaks in the log-log plot.

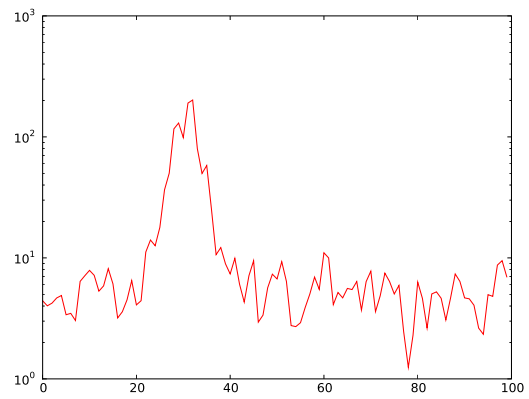
interesting observation here is that oscillations at higher frequencies are much more salient than at lower frequencies in this regime of $\frac{1}{f}$ behavior. This is of course due to the power of the background pink noise declining as frequencies increase. Based on our presumptions, we know that gamma (fast) rhythms are important in attentional binding. We won't pursue this point further, but this builds an intuition that rhythms at high frequencies are important somehow to overcome the background noise.

What happens as these two peaks move toward each other? We create two systems of oscillators, one with bias of 30 Hz and one with 60 Hz oscillations, and gradually move one set of oscillators toward the other. Additionally, we use white noise for simplicity this time, which means uniformly distributed oscillators in the region 0 to 100Hz, with exception of the biased areas. Intuitively, the power spectrum should increase in the overlapped regions as one set moves toward the other. This is confirmed by figure 3.7.

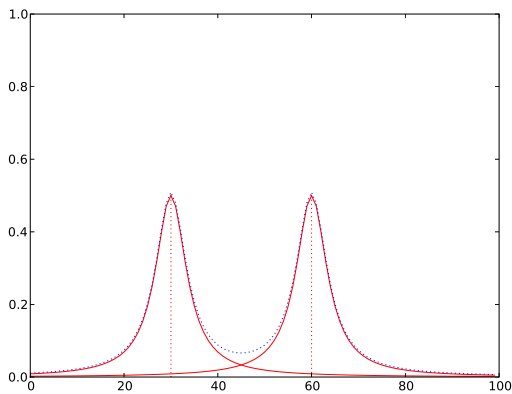
The reason we show this rather obvious example is that ultimately we are after a simple abstract expression that can capture this dynamic. Figures 3.7 (c) and (d) show the so called Lorentzian function at similar peaks as (a) and (b). The Lorentzian function



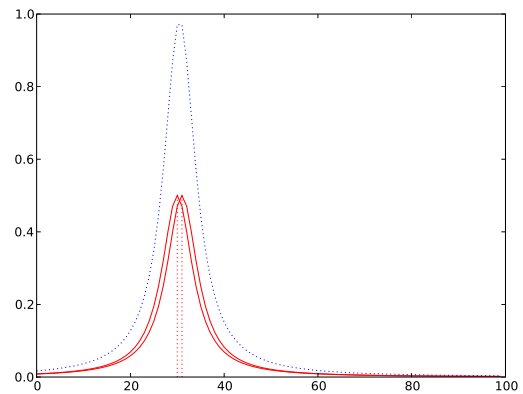
(a) Two oscillator sets at 30 Hz and 60 Hz



(b) Same oscillator sets at 30 Hz and 31 Hz



(c) Lorentzian at 30 Hz and 60 Hz



(d) Lorentzian at 30 Hz and 31 Hz

Figure 3.7: Addition of oscillators. As the two oscillator sets of 30 Hz and 60 Hz gets closer to each other, the power increases in the frequencies that overlap. (a) At start they have little influence upon each other. (b) When they fully overlap, power has doubled in the corresponding region. Graph is logarithmic on the y axis. (c) Lorentzian functions at corresponding frequencies as (a). (d) Lorentzian functions at corresponding frequencies as (b). Blue dotted line is the sum of the two Lorentzians. See text for description of Lorentzians.

is used by physicists as it is the solution to the differential equation describing resonance. We will get back to exactly why this is important after we study connection effects and synchrony between these systems, but suffice for now to say that we use this function to represent the system of oscillators in a single equation. We will also see later how we can use this function to calculate the influence one system has upon another.

The basic Lorentzian function is given by:

$$f(x) = h \frac{k^2}{k^2 + h^2(x - x_0)^2} \quad (3.6)$$

where x_0 is the frequency with highest power, h is the power or height at that point and $f(x)$ is the power at a specified frequency. The parameter k represents the width of the function; a high k value flattens the corresponding graph. The parameter k can roughly be interpreted as the amount of oscillators we have in the system: when expressed this way, the area under the Lorentzian function is $A = k \pi$. This means as k stays constant, and h increases, the graph will get correspondingly thinner as the area stays constant. This fits well with our intuition that as more and more oscillators are synchronized toward some frequency, the power at that frequency increases while it lessens at others, for some fixed number of oscillators. The reason we use this function is that it captures the fact that a population has many different frequencies it oscillates around, and we will use this fuzziness to our advantage.

We will study the Lorentzian in much greater detail later as it will be the foundation on which we build our model. To sum up this section, we have studied a set of independent oscillators and how such a system is an abstraction for a population of neurons. We have introduced the Lorentzian as a function that represents an arbitrarily large set of oscillators in a single equation.

3.4 Partial synchrony

In the previous section we studied a system of independent oscillators, but this is of course a very idealized situation. In practice, all oscillators in such a system would affect all the others. What we are interested in here is whether one set of oscillators can influence another and synchronize at some frequency. Here, we define synchrony as oscillating together at the same frequency.

In a system with a large number of oscillators, determining if the system will synchro-

nize or not seems to be a very hard problem. Luckily for us analytical solutions was found many years ago. Here we will focus on Kuramoto's mean field solution. The history of determining synchrony analytically is wonderfully recounted in Strogatz' Sync [47], which is well worth the read.

As a starting point, we use our generalized oscillator model as described in equation 3.4 and modify it to be affected by the activity of other oscillators:

$$\theta_i(t+1) = \omega_i + \theta_i(t) + \frac{K}{N} \sum_{j=1}^N \sin(\theta_j(t) - \theta_i(t)) \quad i = 1 \dots N \quad (3.7)$$

The additional term represents the influence from the other oscillators. Here K is the mean connection strength. The summation part sums up the differences in phase between all other oscillators, and results in a positive or negative number. As can be induced from the equation, a positive number will speed up the oscillator while a negative number will slow it down in relation to the constant frequency ω . The intuitive part to grasp is that if an oscillator is faster or slower than the mean, it will regulate it's frequency to arrive in sync with the other oscillators.

However, it is not certain that an oscillator is able to change it's speed as much as necessary. If it's natural frequency ω is too far away from the mean, it will not be able to synchronize. This was one of the insights of Kuramoto [46], and thus the concept of partial synchronization was introduced. In a large population, only those oscillators with ω close to the mean will synchronize, while others will be either too slow or too fast. Kuramoto goes on to prove this analytically, which we won't replicate here, but the upshot is that if the distance in frequency is less than the strength of the connection, the oscillator will synchronize: $|\omega_i - \omega_{avg}| < K$. If not, it will desynchronize. Which oscillators will synchronize depends largely on the K value. If K is strong, a few oscillators will not sync, while the rest will bunch together to one large synchronized continent. If K is smaller, multiple islands will form. And if K is close to 0, all oscillators will be desynchronized.

So, what does this mean for us? Let us for arguments sake say we have one system of oscillators A which is influenced by another system of oscillators B by some set of connections between them. Both A and B have initially a low degree of internal synchrony, meaning they have a close to uniform distribution of oscillators. In this case, B will not affect A by any large degree. Contrarily, if B were to have a large degree of internal synchrony, this would create a powerful influence on A . This would be due to the last term in equation 3.7, as a large amount of incoming oscillators would have similar frequencies. The result would be those oscillators able to synchronize toward the new influence would

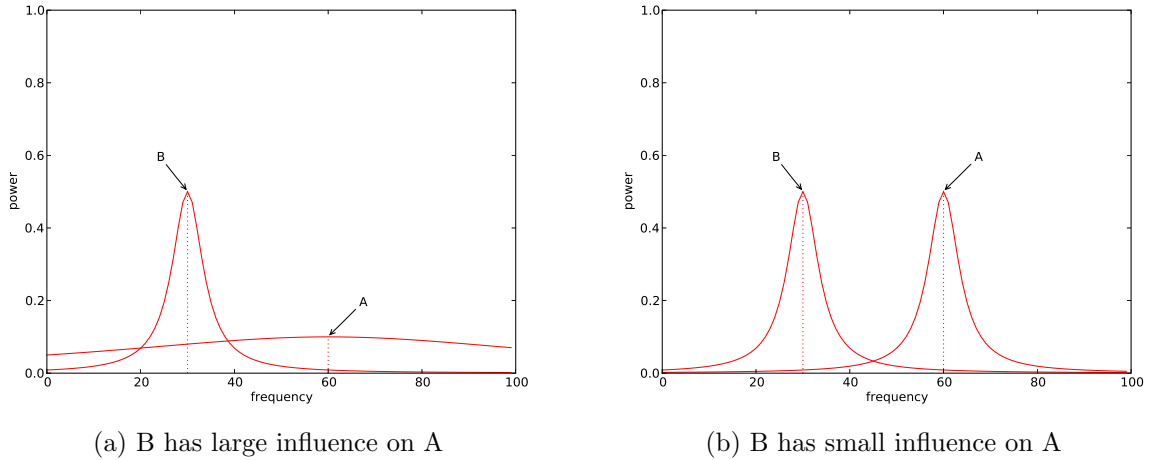


Figure 3.8: Influence of system B on system A. (a) B has large influence on A as A’s oscillators are mostly uniformly distributed. As B has large degree of internal synchrony, it will create a large influence on the oscillators of A, so they would start synchronizing with B. (b) Here, A has a large degree of internal synchrony from before, and most oscillators are preoccupied at the internal synchrony, leaving A less affected.

speed up or slow down. This is visualized in figure 3.8a, where B exerts a large degree of influence over A . A will quickly synchronize with B , because the influence from B will exert the strongest mean field, which the oscillators are drawn toward.

If however A also has a large degree of internal synchrony, many oscillators will be preoccupied at those frequencies, which means B will again have less of an influence. So, even though population B has a strong oscillatory activity, A can oscillate strongly at different frequencies and stay largely unaffected. This is visualized in figure 3.8b, and is a key property we will use in our model.

Finally, if A and B are connected to each other and have a large degree of overlap in frequencies, they will synchronize and resonate on each other. If A exerts strong enough influence on B and vice versa, the last term in equation 3.7 ensures that the influence will increase every time step as more and more oscillators get recruited to the mean frequency. This resonant behavior is also a key property we will use in our model, one which the Lorentzian is well suited for.

To summarize, we employ a sort of population level interpretation of Kuramoto, where the natural frequency is flexible due to the system consisting of many underlying oscillators. As seen, populations can affect each other, and the outcome depends on the current activity. For example, populations can synchronize and resonate together. Also, two populations can oscillate at different frequencies without affecting each other to any large

degree. This is the postulation behind the temporal binding hypothesis: populations that oscillate together are bound to some sort of representation.

These two observations taken together is the core functioning of our model which we will present next.

3.5 The model

By now it should be clear now that our view point is that neural populations can be considered as a system of many oscillators. This is a departure from most other models, which consider single oscillators as a functional unit. Here, the system of oscillators is the functional unit, and we use the Lorentzian function to represent the activity and influence upon other populations. In this section we will describe how we do that. As we are going to be constructing networks of these functional units to perform some specific tasks, we will from now on refer to populations of neurons as nodes in the network.

3.5.1 Model dynamics

First of all, we define that a node at any given time has a sort of mean field that the oscillators are drawn toward. This is in keeping with the Kuramoto model reviewed in the previous section. This gives every node a main frequency band, which we refer to as b . Secondly, the power of this mean field is referred to as a , mainly in reference to the amplitude or general level of activation. This gives the following expression:

$$a(x) = a_0 \frac{k^2}{k^2 + a_0^2(x - b)^2} \quad (3.8)$$

where x is a given frequency, $a(x)$ is the height of the Lorentzian at point x , b is the node's current mean field, a_0 is the height at that point. The height of the Lorentzian represents the power at that given frequency. Figure 3.9a plots this function for a few different values. Notice that for constant k , the graph gets thinner as a increases. Mathematically, this is due to the fact that the area under the function is $A = k\pi$ and thus constant for constant k . For our purposes this fits well with the intuition of a node being a set of oscillators, as more and more oscillators get recruited to the mean field the stronger it gets. In our experiments we treat k as a constant parameter. We recognize that some optimization could be made by varying this parameter, and some experiments were done in this direction, but the gain was not in proportion to the added complexity so this will

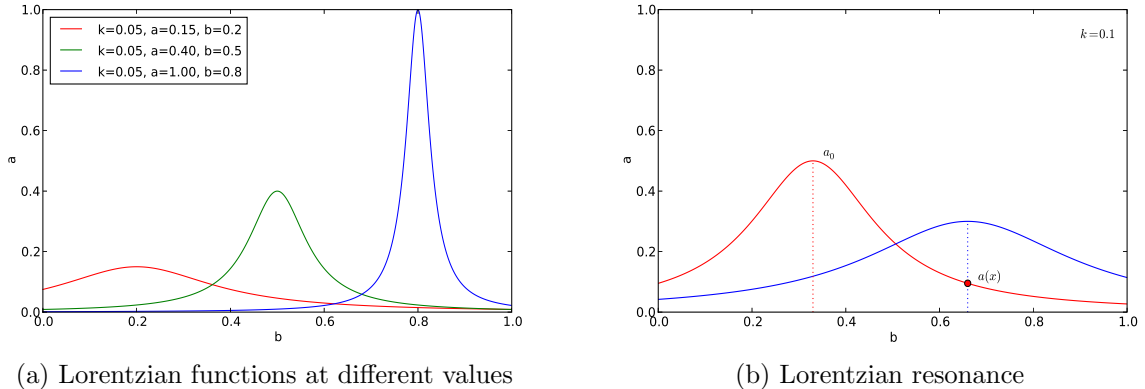


Figure 3.9: Lorentzian functions and resonance. (a) Lorentzians at different values. The k parameter is held constant, and it is worth observing that as the height increases, the resulting graph gets thinner. This is because the area under the expression is held constant. (b) Graphical depiction of how we calculate the resonance of one node on another. The point $a(x)$ is the power of the red node at the mean field of the blue node, and thus a measure for how well red resonates with blue. As these two nodes get closer, this value will increase non-linearly. Or, if red increases and blue does not get closer, this value will actually decrease as red gets thinner.

not be pursued further here.

Also note that $a(x) > 0$ for all x . This can be interpreted that the node has oscillators at every frequency. This is a small drawback as $(x - b)^2$ extends toward infinity, but we take measures to ensure that b and x always stays within reasonable values.

When we connect nodes together, they will have some influence on each other. The effect of this influence can be divided in two, one affecting a which we define as *resonance*, and one affecting b which we define as *synchronization*. Figure 3.9b shows graphically how we determine the resonance effect a . The node shown in blue color is resonantly affected by the red node's value $a(x)$ where x is the frequency band of the blue node. This is simple to calculate from the Lorentzian of the red node. The only necessary values are the red node's a_0 value and the distance between the b values of the nodes. Thus, the values a and b are not independent of each other.

In general, a node's a value is simply the sum of all incoming nodes' $a(x)$ values. In addition, we introduce a weight to all connections. In traditional artificial neural networks, this weight represents the synaptic efficiency between two neurons. Here, the nodes are populations of thousands of neurons and the weight has to be interpreted differently. While having the same effect, we interpret weights here as how well the neural pathway between two populations faithfully transmits the oscillations. While being more abstract,

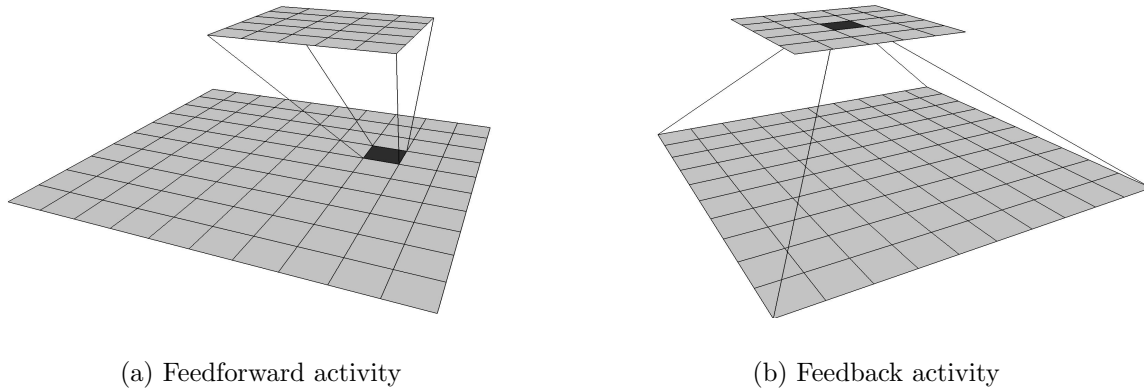


Figure 3.10: Feed-forward and feedback network activity. The bottom layer is the input layer, and the top layer is the output layer. (a) Nodes in the bottom layer project their activity to the top layer. (b) Nodes in the top layer feed their activity back to the lower layer. There are no lateral connections in the model.

it can be thought of as the collection of neurons employed to transmit the oscillations.

The red node in figure 3.9b will also have an effect on the blue node's frequency band, b . As we saw in the previous section, a strong mean field influence will attract the oscillators, thus in effect moving the b value toward this influence. As the distance between the b values of these nodes decrease, it is easy to see that they will mutually exert more and more influence on each other. Now, in our networks any given node will be influenced by many other nodes. It is conceivable that strong influences will come from many inputs, and the oscillators in the node will attach to the one which is closest and strongest. This means that a given node realistically can have many peaks. Also, as we started doing experiments on this system, it quickly became evident that having a single b value didn't give the results we wanted, because of this lacking ability of differentiating between influences. The real problem was that a single node could have many inputs, but mapping these to a 1-dimensional frequency spectrum became very problematic.

So, instead of having a single b value for every node, we employ a mechanism used in fuzzy systems. In fuzzy systems, the membership of an object to a class is not binary but rather a degree. An object can then be part of many classes. For example, a given temperature can be both warm and hot; there is no fixed point at where this transition occurs. Implemented in our system, this means we track the frequency distance values d_{ij} between every connected node i and j , instead of calculating them as $(x - b)^2$. Our choice of distance function ensures that we only synchronize strongly to one node at a time.

Before we present our model, we need to introduce the network structure these nodes

will be used in. The network is designed as follows.

1. A bottom layer consists of a set of nodes which receive input from some pattern. The output amplitude a of these nodes is a function of their inputs and feedback from top layer nodes. The distances d to all top level nodes is initially set to 1.
2. A top layer receive inputs from the bottom layer through feed-forward connections. Here, the amplitude a is calculated as the degree of resonance with the nodes from the bottom layer.
3. The top layer sends feedback to the bottom layer, which is used to calculate new distance values d according to the most activated nodes in the top layer. This subsequently results in increased resonance at the next time step. Also, a slight resonance affects the bottom level a values.

The rationale for this design is that the network should arrive at a winning node through the dynamics in the system. The bottom level nodes are considered feature nodes; whenever a feature in the pattern is present, the corresponding node is activated. This means we need as many bottom level nodes as there are potential features. The top level nodes are considered category nodes; when they are activated the corresponding category is present. The system is initially unsure which category is the correct one, especially in the presence of noisy or composited patterns, such as we have seen earlier in the Rosenblatt superposition catastrophe. But as top level nodes get activated by bottom level nodes according to their weights, and the bottom level nodes get modulated by the most active top level nodes, the network settles in to the most likely categories. We will simulate exactly how this works in the next chapter.

So, the resulting equation set for our system is as follows:

$$a_i = \frac{1}{N_l} \sum_{j \in N_l} W_{ij} a_j r_{ij} + \beta \sum_{j \in N_h} W_{ij} a_j r_{ji} + I_{ext} \quad (3.9)$$

$$r_{ij} = \frac{k^2}{k^2 + a_j^2 d_{ij}^2} \quad (3.10)$$

$$d_{ij}^2 = 2 \left(1 - \frac{W_{ij} a_i}{\sqrt{\sum_{n \in N_h} (W_{nj} a_n)^2}} \right) \quad (3.11)$$

In many ways, these 3 equations are the main result of this thesis. After uncountable revisions to get a dynamic that works, this set is the one we arrived at that gave the

best results. It is still not perfect, but works very well. Equation 3.9 is the *activation* function, equation 3.10 the *resonance* function, and equation 3.11 the *distance* function. The equations are designed to be modular, so the resonance function and the distance function can be exchanged with other compatible functions.

Equation 3.9 is the function that governs general activation of a node. The first term simply sums up all resonant influences from a lower layer and normalizes it. We normalize here instead of normalizing the weights during training, mainly to keep the learning rule simpler. N_l is the set of nodes at a lower level connected to i , W_{ij} is the weight between these nodes, a_j is the activity of the lower node and r_{ij} is how well the nodes are resonating, represented by the resonance function. The second term is similar to the first, but concerns resonant activity from nodes at higher levels. Here, N_h is the set of nodes at the higher level connected to i . We do not normalize this term however, but multiply by a parameter β , $0 < \beta < 1$ instead. The point is that top-down influence is mainly modulatory as we saw in section 2.3 and should not activate lower nodes much. By setting this parameter to values less than 1 we also ensure that a higher and lower node do not resonate their activity to infinity. Thus, this parameter functions as a limit and is usually set at around 0.5. Notice that the resonance function switches order in this term; this is so we can differentiate between feed-forward and feedback activity.

The final term, I_{ext} , is for adding any external input, primarily for input nodes. However, I_{ext} is also used on output nodes for learning. We will introduce the learning rule later, but using this parameter we can force a category node to be activated so that its incoming weights can be adjusted for the given input pattern.

Equation 3.10 is the Lorentzian resonance function as we have seen multiple times so far. The resonance between two nodes, r_{ij} is dependent on the parameter k , which is a parameter of the model, the activity of the lower node a_j and the distance in frequency between them. This function acknowledges that feed-forward connections are the ones that drive the network.

Equation 3.11 is the function that calculates the distance in frequency between upper layer and lower layer nodes. While looking complex, it is simply the result of a vector distance. The idea is that we treat the activity a of every incoming higher node as a vector orthogonal to each other. This means creating a vector representing the weighted a value for each input, summing them to a single vector, then normalize it to length 1. We then calculate the resulting distance to this hyper dimensional point from the unit vector in the dimension corresponding to an input, and use this as the distance between each node. This approach ensures that a lower level node can synchronize strongly with

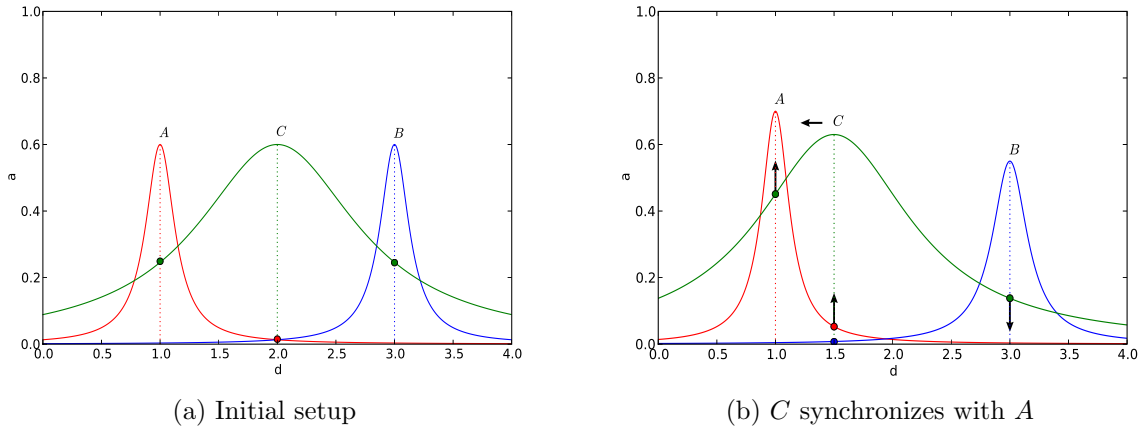


Figure 3.11: Simple feed-forward and feedback dynamic. A and B are top level nodes and C is a bottom level node. The parameter k is 0.1 in the top level, and 0.5 in the bottom level. (a) Initial position. Green dots are feed-forward driving activity and represent the r_{ij} resonance values. (b) Every time step, as C for instance moves toward A , A will receive increased resonant activity, while B will receive decreased activity. At start, A and B both have relatively small feedback resonant influence on C (red and blue dots, representing r_{ji}) which is regulated by the β parameter. As distances decrease C will increase it's a value due to increased feedback resonance (here $\beta = 0.5$). The result is C strongly synchronizing with A , and B receiving much less input from C .

only one higher level node.

As mentioned, these functions are designed to be modular, and an alternative to the distance function would be to simply calculate a weighted mean. However, that had drawbacks in not representing the independencies of the nodes very well. This approach with orthogonalizing the nodes instead is computationally equivalent, and bottom level nodes are able to better distinguish top level nodes.

Figure 3.11 shows a simple example of these dynamics between nodes. C is a bottom level node, while A and B are top level nodes. We initialize every distance between top and bottom nodes to $d_{ij} = 1$, thus all bottom nodes initially give equal input to every top node. The k parameter is usually differentiated between top nodes and bottom nodes as can be seen in the diagram. Typically, $k_{top} = 0.1$ and $k_{bottom} = 0.5$, as they are here. Initially, C has equal influence on A and B due to equal distance. The feedback resonance is negligible.

Let us for arguments sake say that A has the stronger influence on C . This results in $d_{CA} < d_{CB}$, and as the distance between A and C decreases, A will increase activity due to higher resonant effect, and B will receive less resonant input from C as the distance

increases. Additionally, C will gain increasing resonant feedback and eventually synchronize totally with A . B will at that point receive almost no input from C at all, even if the weights are strong and activity high.

This system solves both the separation and segmentation problem described earlier. Separation means disambiguating input patterns. In the case of compositional inputs where two or more patterns are presented at the same time, the dynamics explained here will be able to separate these patterns fairly well if previous experience has exposed the network to these patterns individually. Segmentation entails synchronizing the input node to the category it represents. In our system, segmentation is a necessary step enabling the separation of patterns. We will explore both separation and segmentation much more in the next chapter. A short example of this model with runnable code in Python is included in appendix A.

To recap, our model is governed by three equations: the activation function, the resonance function and the distance function. We particularly saw that rather than have a single b variable, each node has a vector of d_{ij} representing the frequency distances to each connected node and thus the degree of membership to that node. By employing this mechanism common in fuzzy systems, and indeed by defining our nodes as a population of oscillators which can oscillate at many frequencies simultaneously, we call our model simply *Fuzzy Oscillations*.

3.5.2 Model learning

Learning in our model is relatively simple, and follows the Hebbian tradition. We slightly paraphrase Hebb, and sum up the learning principle with “oscillate together, wire together”. This is more in tune with what we know of STDP if we constrain the oscillations to be in the region of gamma rhythms. We express this as follows:

$$W_{ij} = \lambda (a_j - W_{ij}) a_i^2 r_{ij} \quad (3.12)$$

In equation 3.12, λ is the learning rate. The second factor represents the direction we are tuning the weight. If the activity of the lower level node a_j is high, the weight is tuned up, and if a_j is low, the weight is tuned down. The activity of the top layer node is represented by a_i^2 . The last factor r_{ij} is the resonance function and represents how much the nodes are synchronized. So, maximal positive learning occurs when both a_j and a_i are high and in sync. This reinforces the strength between them. If they are not in sync,

little learning will occur between them. If a_j is low while a_i is high, the link will weaken, as it does not contribute to the activation. If a_i is low, no learning occurs at all. The importance of the activation of a_i is emphasized by squaring it.

In our experiments we differentiate between supervised and unsupervised learning. Supervised learning occurs when the correct category to an input pattern is given by some instructor and the network should learn to respond with that category for the input pattern. When employing unsupervised learning, the network typically has to learn to represent the categories on its own for any given input pattern.

As our focus in this thesis is the actual dynamics of the system, we mostly employ supervised learning. We have already hinted at how to do that. If all weights are initialized randomly at start, we can guarantee selection of a particular output node by adding artificial input to that node, thus keeping it highly activated. This is done by setting $I_{ext} = 1$. The input nodes will quickly synchronize with the highly activated output node, and equation 3.12 ensures that weights are tuned accordingly. When employing this type of learning, we use a high learning rate, for example $\lambda > 1$. We can train the network very rapidly using this method, usually in under 20 time steps per pattern. We reinitialize the network between each pattern presentation with exception of the weights.

Unsupervised learning is similar except that there is no instructor to force input to a category. The important thing that the network needs to do is avoid interference from other patterns when selecting a particular node as representative for a category. This gets increasingly difficult as more and more of the available top level nodes gets dedicated toward different patterns due to capacity problems. As we shall see later, our system as it currently stands is not able to do this perfectly, and is an avenue of research outside the topic of this thesis. In this discussion section we will also present some alternative approaches.

3.5.3 Experiment environment

In the next chapter we employ three different scenarios to test our model. The scenarios are described in full there, as there are some differences between them.

As a preparation to perform our experiments, we developed an entire Python IDE application, with one goal: *never lose experimental results again*. Previous experience dictated that it is difficult to keep results when experimenting on many different parameter combinations. After losing many results in the past and wondering what the parameters were that gave excellent results, we created a Python development environment that

automatically saves and dates each experiment. One can go back and look at both results and the settings of individual experiments. While it sounds like a large undertaking, most of the tools already existed in various Python libraries. We used wxPython as graphical user interface, which produced a rather fully featured IDE, such as code coloring, integrated python shell, and even code completion.

All experiments in this thesis was done using this IDE and thus written in Python. A few libraries however was written in C++ and imported in to the Python environment using boost.python as a bridge. The Izhikevich neuron environment is an example of this, as it is particularly computationally intensive.

Chapter 4

Experiments and results

In the previous chapter we developed our *Fuzzy Oscillations* model. In this chapter we delve in to actual experiments and the results obtained. The experiments we perform here represent the very first steps of exploring the basics of our model.

The first scenario is a very simple setup designed to impart an intuitive understanding of how the system works. We will not use any learning here, but step through the algorithm to showcase the dynamics. Network size is only 4 to 6 nodes in total.

The second scenario focuses on simple image recognition. But the goal is not to create the best image recognizer, but rather experiment with training on individual patterns and testing with pairs of simultaneous patterns. Here, we use an input region of an 8 by 8 2-dimensional grid, where each pixel represents a feature. We have 16 different patterns, and the goal is to learn to separate and segment the images. We show how well it separates between patterns, and how well it copes under noisy data. Network size is 64 input nodes and 16 category nodes.

The goal in the third scenario is, as before, to learn patterns presented independently and then learn to separate patterns that are presented together, but this time to compare our results with other systems. The network is much larger, and we really put the dynamics of the model to a test when considering many simultaneous patterns. Network size is 512 input nodes and 26 category nodes.

4.1 Model fundamentals

This first scenario revolves around simple composition of patterns. We do not use any form of learning yet, as we initially want to impart a feel for the dynamics of our network before getting in to more complex issues. As such, we use binary connections where a link is implicitly set to have strength 1, where a non-connection has strength 0. This scenario is borrowed from [2].

4.1.1 Simple composition of features

The first experiment we perform is the simplest example of the networks. Suppose we have two category nodes that represent the letters P and R. R shares some patterns with P. We define two possible input features, 'P' and '\'. We connect the network such that both category nodes P and R is connected with input pattern 'P', but category node R is additionally connected with the independent input pattern '\'. Figure 4.1a shows the connections in this network.

The interesting question to ask is what happens when all features are presented at the same time? Traditional neural networks would answer that by having both categories firing simultaneously. However, here we have the case where one category subsumes or encompasses another. Due to additional evidence and composition of features, we know that when both 'P' and '\' are present the correct category is R. Figures 4.1b and 4.1c show how our system solves this problem through the dynamics of the network. The smaller category P is unlikely given features specific only to the larger R. That is why P loses in competition with R in this case. If only the 'P' feature was present, P would win over R.

The settings in this experiment were $\beta = 0.0$ meaning we disabled feedback resonance to bottom layer nodes. The k parameter was set to default. Figure 4.1c shows the values for each time step in the simulation. At start, an input of 1 is added to the input nodes, and frequency distances to the top level nodes is set at 1. At $t = 1$, we see that the '\' feature has already synchronized with the R node as it has no other influences. The 'P' node on the other hand is more uncertain and it still belongs equally to P and R category nodes. As such, the distance values d_{ij} represent the fuzzy membership values; input nodes are resonating with various degree with all output nodes. In the next time step, R has gained more activation as it is now resonating strongly with '\'. It then exerts more influence on input node 'P', and the frequency distance decreases. This results in less resonant activity with P which subsequently loses activation. This is similar as in figure

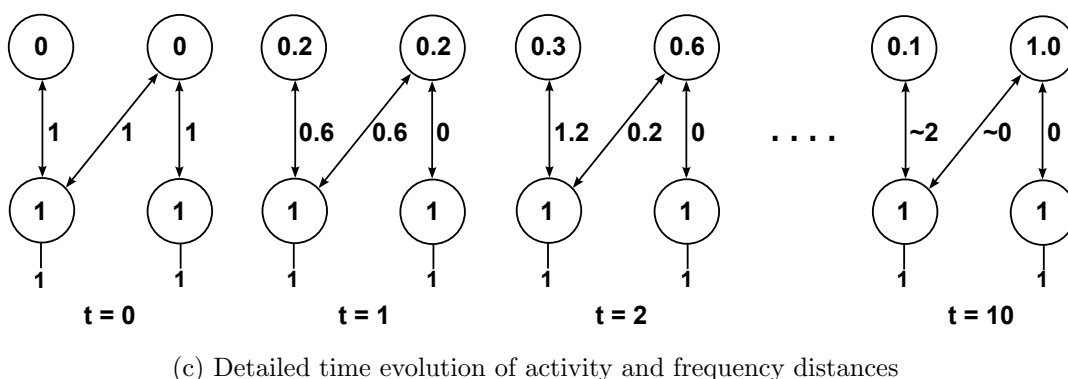
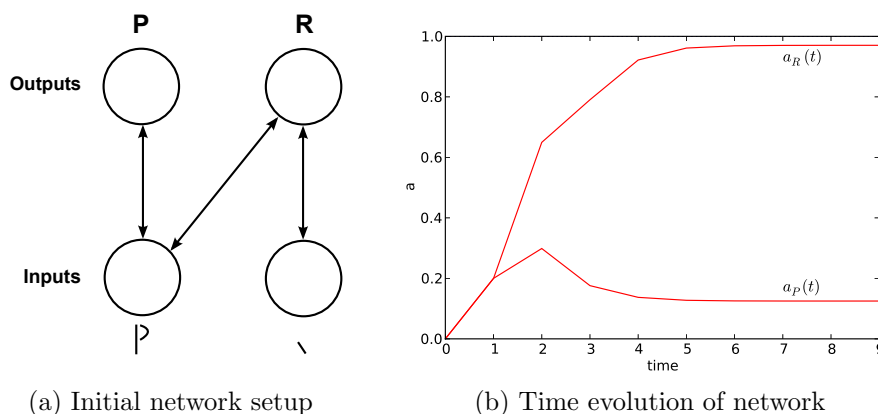



Figure 4.1: Simple P-R scenario, where all inputs are on simultaneously. (a) Initial setup of network. Top layer nodes represent categories P and R. Bottom layer nodes receive the input pattern. The missing connection in the diagram is cut away as it has weight 0. The rest have weights 1. (b) Resulting activation of top layer nodes as network evolves. While initially equal, the added evidence of '∖' enables R to gain more influence over the 'P' input, thus strengthening R while weakening P. The network settles after very few time steps. (c) A numerical simulation of (b), showing how activity and frequency distances evolve over time. Numbers inside the circles represent the activity a , numbers connected to the lines represent frequency distances d_{ij}^2 . As is evident, the right input node immediately synchronizes with the output node R as it has no other influence, and the frequency distance falls to 0. The gained input from this node allows R to gain higher activity, thus exerting more influence over the left input node. In these diagrams we see how there is competition between output nodes without any explicit lateral inhibitory connections.

3.11 where this dynamic is explained in better detail. The network stabilizes after a few time steps, and R emerges victoriously. Note that activation of P does not go down to 0; this is due to our distance function being bound by a maximal distance. One can interpret this as there is still some small probability that P is the correct category, especially in noisy environments.

This small example clarifies what we mean by separation and segmentation. The clear separation of patterns is seen in figure 4.1b, where one representation wins over the other. Even though all inputs were present to the network, it successfully disambiguated the features. Segmentation can be seen in that both input nodes are synchronized with the winning category after the network has settled. This is evident in the small distances seen in figure 4.1c.

4.1.2 Avoiding binding errors

In this experiment we evaluate three simultaneously presented features, by expanding slightly on the previous experiment. Here, we have three potential features: $\circ \circ$, $-$ and \lrcorner . The first is categorized as wheels, the first and second together categorizes as barbell and the second and third is a car chassis. Essentially, the network structure is a simple expansion of the previous experiment, and the network structure can be seen in figure 4.2a. The goal of this experiment is to demonstrate how the network dynamics takes all evidence into consideration when choosing categories, thus avoiding binding errors.

The parameters are set exactly as in the previous experiment. Figure 4.2b shows how the network evolves over time. At $t = 1$, we see that all categories initially activate equally. The chassis gains an early lead as it has a feature which synchronizes immediately, and thus gains a resonant increase in activity. This has the effect of exerting more influence on the $-$ feature, thus synchronizing it with chassis rather than with barbell. Barbell loses activation because of this. On the other side, wheels and barbell initially exert equal influence over the $\circ \circ$ feature. As barbell loses activation due to other present features, the wheels category gains the advantage and that feature eventually synchronizes with it, leaving the barbell with little to no activation. One could visualize another layer on top which would recognize the combined features of the winning categories of wheels and chassis as a car .

Given an image of a car with all features simultaneously activated, choosing the barbell would technically be a correct categorization since all its inputs are present. However, this would be the equivalent of a binding error as it ignores the context and additional

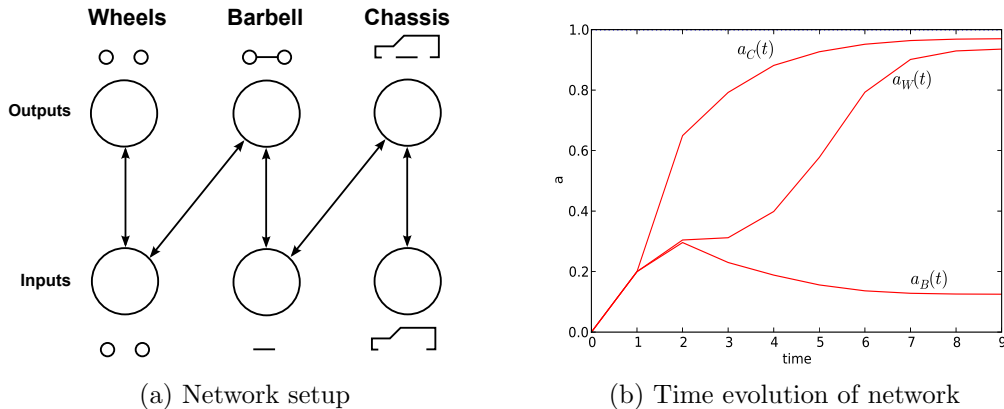


Figure 4.2: Wheels-Barbell-Chassis scenario, where all inputs are on simultaneously. (a) Initial setup of network. As before, all visible links have weight 1. (b) Resulting activation of top level nodes as network evolves. Chassis category wins as it has a node that directly synchronizes with it, as in the previous example. This means that the $-$ feature will get influenced more strongly by chassis than by barbell, and barbell will lose activation. The consequence is that the $\circ \circ$ feature now will be more influenced by the wheels category, which will increase activation and decrease barbell even more. The winning categories are thus wheels and chassis, and they both get highly activated as they do not compete over any features between them.

information in the scene. This experiment clearly shows the effect and importance of segmentation. As categories are recognized and their inputs synchronize with the correct category, it becomes easier to disambiguate between other patterns as they compete over the remaining features. *This is a key point in our system*, and we will discuss this more in the next chapter. We have included a short code snippet that runs this example in appendix A.

These two experiments were borrowed from Tsvi Achler’s regulatory feedback system described in [2]. We have previously reviewed his system, and he gets similar results as we do with regard to pattern separation. However, his system does not have any concept of segmentation, but is also computationally much simpler than our system. Also, he does not employ any learning mechanism, which we do from now on.

4.2 Simple image recognition

In this second scenario, we study simple input patterns from static images. The method used in this scenario is to train the network on single patterns, and test it on composited patterns created by superimposing pairs of the static images. We don’t study any form

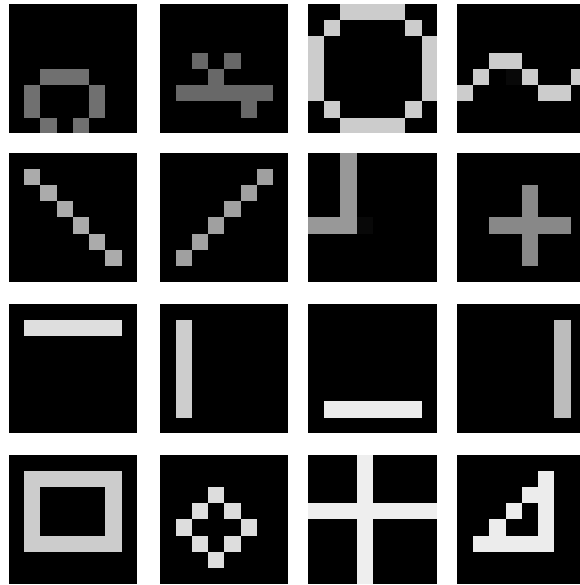


Figure 4.3: The 16 input images used for simple image recognition. The network is trained with a single image at a time, and tested with compositions of multiple patterns.

of variance (scaling, rotation or translation) of the input patterns, that is subject for a later study. The goal of this scenario is to measure how well these networks separate and segment patterns when only previously exposed to single patterns. This scenario is borrowed in part from Rao et al. [36], and we will compare it a bit later.

We employ supervised learning in this experiment. We create a network of 64 input nodes and 16 output nodes. The weights are initialized to a Gaussian distribution around 0.5, so that output nodes can initially be distinguished. The input nodes receive input from the 8x8 patterns presented to the system. Figure 4.3 shows the simple input images used in this experiment. We use all standard parameters, and set the learning rate to $\lambda = 0.5$. For each input pattern, we select an output node and set its input $I_{ext} = 1$. This ensures that this node will win over any other. The input nodes for the pattern synchronizes rapidly with the output node, and weights from active input nodes are strengthened, while weights from inactive nodes are weakened. Using this learning rate, we can teach the network quite rapidly. Each pattern is exposed to the network for only 20 time steps, and a single pass over all input patterns is sufficient to train the network.

After learning, the network is ready to be tested, and the learning rate is simply set to $\lambda = 0$. Thus, we have no external algorithm like back propagation to teach the network. A few words about the learning rate; if it is set too high the network tends to forget too quickly as the first patterns tend to weaken. This is again due to the distance function which bounds the maximal distance, so there will always be some tuning between all

nodes. So, as it currently stands, the learning rate has to be balanced against the number of time steps each pattern is exposed. An improvement of this will be considered in later studies, but it fits our purposes here for now.

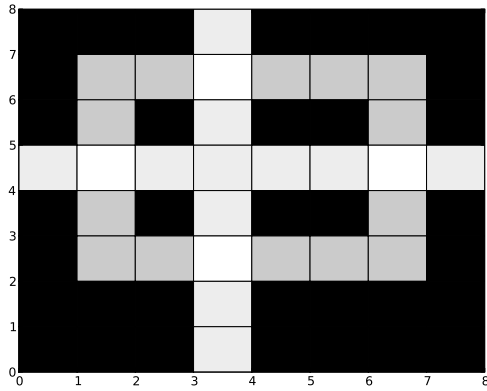
We test the network by creating a set of test images by superimposing pairs of input images. This results in $\binom{16}{2} = 120$ test patterns. As it turned out, it wasn't particularly difficult for the network to separate between these pairs of inputs and did so with 100% accuracy all the time. As evident in figure 4.3, the input images are fairly different from each other, and the network had no problem in identifying each single pattern when presented with pairs of patterns simultaneously. In the next section we will do a more quantitative study of what happens when we increase to more simultaneous patterns.

So, our fuzzy oscillations model was able to solve the original problem of feature integration and segmentation in this case with 100% correctness, even though it was trained on single presentations of patterns. This means the network was able to move outside of its training set and, more generally, correctly classify patterns very different than the ones it was presented with. This is a departure from feed-forward classification networks, where such compositions must explicitly be trained for. Indeed, it is the dynamics of the model that makes this possible, and we will delve in to this a bit later.

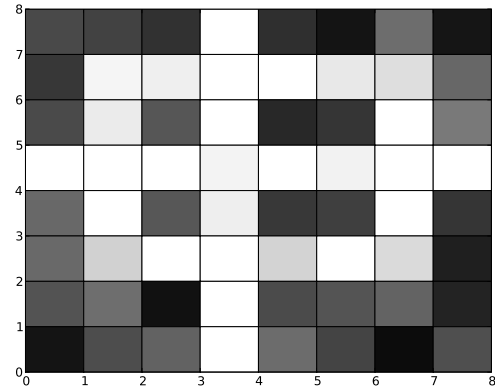
Now, the strong point of traditional classifiers such as ANNs is their ability to generalize. This means being able to classify patterns even though it is not exactly like the stored prototype. We wanted to know how well our model is able to generalize, and we rephrased that question to how robust it is in the presence of noise. When testing on the recall ability of single patterns, the network showed considerable immunity to noise, being able to recall with 95% accuracy even with 50% noise. As this isn't very interesting to our original question, we go on to test the recall ability of multiple patterns in the presence of noise.

So we investigated the sensitivity of the network with respect to additive noise. We trained the network with single patterns without any noise. Increasing amounts of noise was then added to each test pattern of two simultaneous images, and network performance was measured. Noise was added to each pixel of the test pattern by drawing a random number from a uniform distribution between 0 and 1 multiplied by a noise fraction increasing from 0% to 50% in 5% increments. To create reasonable statistical evidence, 100 trials was used for each noise level. Each trial presented all 120 noisy test patterns to the network. For each pattern presented, we recorded the separation and segmentation accuracy.

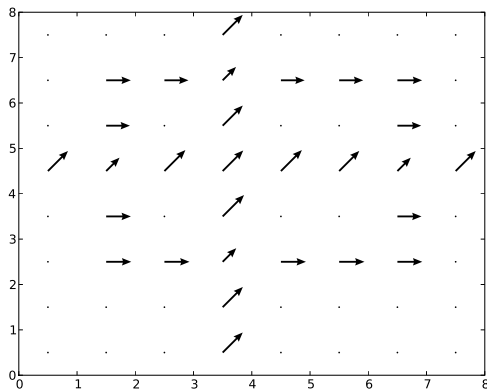
Separation accuracy was calculated by finding the most activated output nodes after



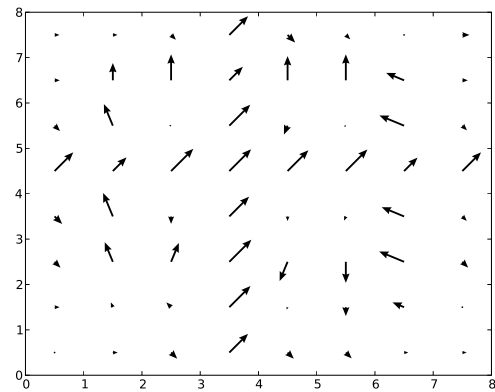
(a) Two superimposed images



(b) 30% added noise



(c) Segmentation field of (a)



(d) Segmentation field of (b)

Figure 4.4: Compositing patterns with noise. The network is tested using two overlapping images, and the task is to separate and segment the images. (a) Compositing image created by superimposing two input images. (b) The same image with 30% additive uniform noise. (c) Synchronization of input nodes. Arrows that point in the same direction represent nodes that are synchronized with the same output node. Arrow length is relative activity of that input node. Notice that the 4 overlapping nodes have a shorter arrow, which means competition for that node. (d) Synchronization of input nodes with 30% noise. Here, the square has not been recognized, and the nodes are synchronized with different output nodes. The cross however was correctly identified in this scene, and the corresponding input nodes are synchronized with the correct output node.

20 time steps. We employed a strict accuracy measure, which meant that we only noted a trial as a success if both winning nodes were the correct output categories. If one or zero of the winning nodes was a correct categorization, we noted the trial as a separation failure. Segmentation accuracy was defined as the ratio of input nodes that synchronized with a correct output node.

Figure 4.4 shows how the test images were created. Figure 4.4a is a simple superposition of two input images. Pixel intensity values were simply added, and clamped to a maximum. Noise was then added, and figure 4.4b shows the same pattern with 30% uniform noise added. A graphical depiction of resulting segmentation is shown in figures 4.4c and 4.4d. In these arrow field plots, each arrow represents an input node. The direction the arrow is pointing represents the output node it is most synchronized with, and the length of the arrow is the activation a of the node. In the previous example we used a β value of 0, but here we set it to $\beta = 0.5$ as the standard value, which means that the input nodes are allowed to resonate with output nodes and increase their activation a . Nodes that don't resonate well with their output nodes thus have shorter arrows.

The resulting segmentation of input nodes for a composited image without noise is shown in figure 4.4c. The segmentation is perfect, and the overlapping nodes of the square and cross pattern has shorter arrows, which means it is not strongly synchronized. Thus there is a bit of uncertainty in those nodes, but the cross has a slight edge. Figure 4.4d on the other hand shows that noise can have serious negative impact on both separation and segmentation. Here, the square has not been recognized at all due to noise, and the input nodes are synchronized with various other input patterns, such as simple bars. The cross however was identified, which can be seen as the input nodes are synchronized together. This particular case would be marked as a separation failure, but would score moderately on segmentation accuracy as many of the nodes are synchronized with a correct output node.

The results of this experiment is shown in figure 4.5. At 0% noise, the network is always able to correctly separate and segment pairs of the input images. The performance naturally degrades as more and more noise is added, but handles reasonably well up to 25-30% noise. Both the separation and segmentation scores measure classification performance of the network, but it is interesting to see how the segmentation accuracy continues to be reasonable even at a high degree of noise. This shows that our networks do have a good degree of fault tolerance, and thus being able to generalize well. In later studies, we will go on to study this in form of how well the network generalizes in the face of scale, rotational and translational variability. Such variability does not give any meaning in this particular experiment however, as exact pixels are taken as features.

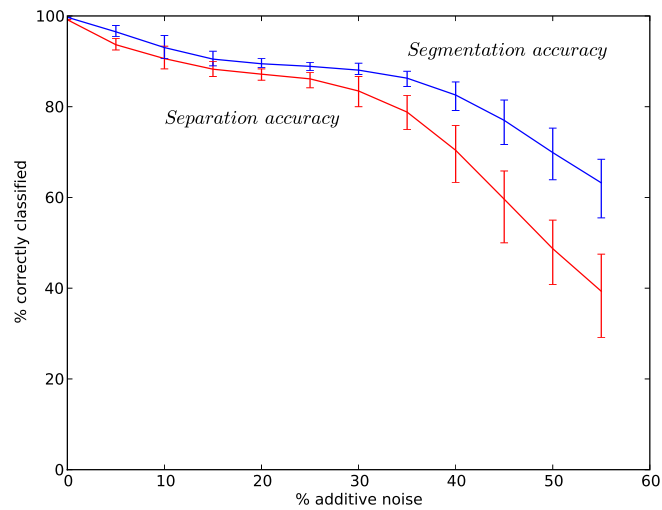


Figure 4.5: Separation and segmentation accuracy in the presence of noise. Network copes reasonably well until added noise is around 25-30%. 100 trials were used to generate each point in the graph.

Figure 4.4b shows this point. Humans can easily see through the noise to see the two patterns it is composed of. But much of that is due to the contrast and edge detection mechanisms humans have early in visual processing. Indeed, such preprocessing is common in machine vision as well [44]. This particular point is important here, and fortifies the position we have that our system is more suitable for higher level cognitive processing. We fully expect a great deal of improvement in image recognition by building our system on top of lower level feature detectors, however that is outside the scope of the current paper. One of our lesser agendas in this thesis is that one needs to use the correct tool for the level one is working at. Our firm position is that low level feature detection is done best by low level tools such as spiking neural networks, but cognitive processing needs higher level abstractions, such as our system. We will discuss this more in the next chapter.

As mentioned, we borrowed this scenario from Rao et al.'s [36] in which we feel misses this point. They do not discuss this however, and we will not pursue that further. In that paper they use oscillatory units and phase synchronization to perform separation and segmentation. We have briefly reviewed his system earlier, and seen that their system is very computationally intensive, and takes a long time to settle. Also, our results are initially superior to theirs as we are able to get perfect separation and segmentation, they never top 80%. However, this is not really comparable: we use supervised learning, while their system is based on unsupervised learning. In fact, when trying unsupervised

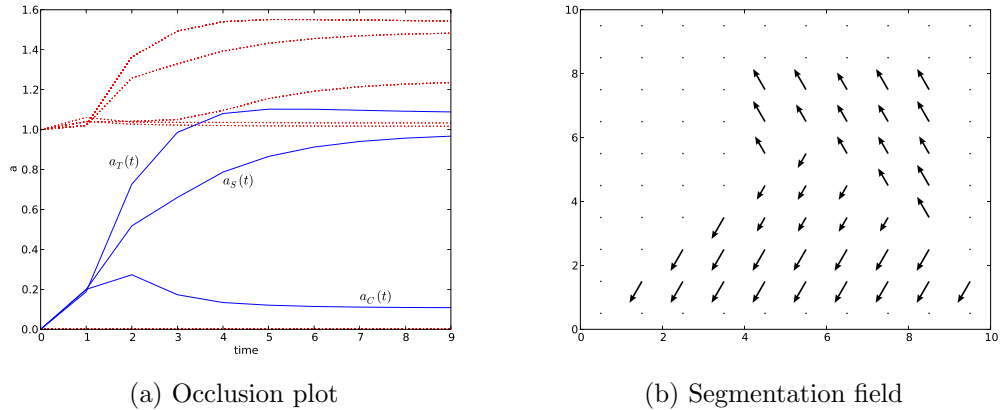


Figure 4.6: Segmentation in an occluded scene. (a) The evolution of the network over time. The red dotted lines represent the activity a of the input nodes. Here, $\beta = 0.5$ and we clearly see the effect of feedback resonance activating the input nodes more than the input itself. The input nodes that don't increase much in a are nodes that are in competition. (b) Resulting segmentation of scene. Again, shorter arrows means nodes that are in competition.

learning, our system routinely under performs, and the best we were able to achieve was around 70% separation accuracy.

We traced the problem to how we normalize the input in equation 3.9. In unsupervised learning, all weights are initialized similarly as with supervised learning to a Gaussian distribution around 0.5. It turned out that new patterns would sometimes rather overwrite a previously learned pattern than select an uninitialized output node. This can be seen as a question of capacity, in that we were never able to fully utilize all possible output nodes when using unsupervised learning. However, since we were able to successfully use all output nodes in supervised learning, this capacity problem does not come from the network itself, but rather from interference with already stored patterns. So this is a different problem than Hopfield's (in)famous empirical proof in [23], where he showed that he could not accurately store more than $0.15N$ patterns, where N is the number of nodes in his Hopfield net. The study of this is outside the scope of this paper however, and we will discuss this and alternatives in the next chapter.

Finally, in this scenario we have studied for the most part images superimposed on one another. This rather constructed scenario was used because it involves both multiple independent images in the same scene as well as partially occluded objects. To better demonstrate object occlusion, we have added figure 4.6. This very simple example has been trained on a circle, a square and a triangle, and tested on an occluded scene where the circle is not visible, and the triangle partially occludes the square. Figure 4.6a shows

how the triangle and square categories both win. The red dotted lines represent the input nodes. While we don't show the scene itself here, figure 4.6b shows the resulting segmentation field where one can see the triangle occluding the square.

4.3 Performance comparison

In the previous section we mentioned the difference between using direct pixel values in an image as features and the use of specific feature detectors to capture the visual scene. In this scenario we use a simple bag-of-features type feature extractor which is applied to all stimuli in the visual field. The detector is inspired by feature extraction methods found in the primary visual cortex [49], but hugely simplified. The exact biological plausibility is not the point here however, but to create a complex enough environment to compare our system with other methods. This scenario is built upon the first scenario we used by Achler [2].

The task is again to train the different systems on single presentations of a pattern, and test them on multiple simultaneous patterns. Instead of directly compositing them on top of each other as in the previous section, we can think of a visual field consisting of multiple images next to each other. We employ a feature extractor which is basically a 3x3 sliding window which slides over the entire visual field. This window records every pattern that is activated within itself. Each such pattern is a single feature that subsequently activates an input node specific for that feature. So, even though the patterns are distributed spatially, we do not use this spatial information for our purposes here. Of course, much better results could be had by using this spatial information, but here we wish to focus on the dynamics of the model and especially how it solves ambiguous patterns.

This scenario involves character recognition, and we have one 5x5 image for each character in the alphabet. A set of characters are inserted on the visual field such as they do not overlap and have at least 2 pixels space between them so they do not interfere with each others feature extraction. The sliding window then covers the visual field which produces a set of activated features. Figure 4.7a depicts this graphically.

The 3x3 sliding window can distinguish up to $2^9 = 512$ different patterns, so our input space consists of 512 nodes. For simplicity, we define that these feature detectors are binary: either the feature is activated or it is not. This means we do not count multiple repetitions of each pattern, even though this could help us distinguish composited patterns better. However, our goal here is not to create the best character recognizer, but rather compare different systems against each other.

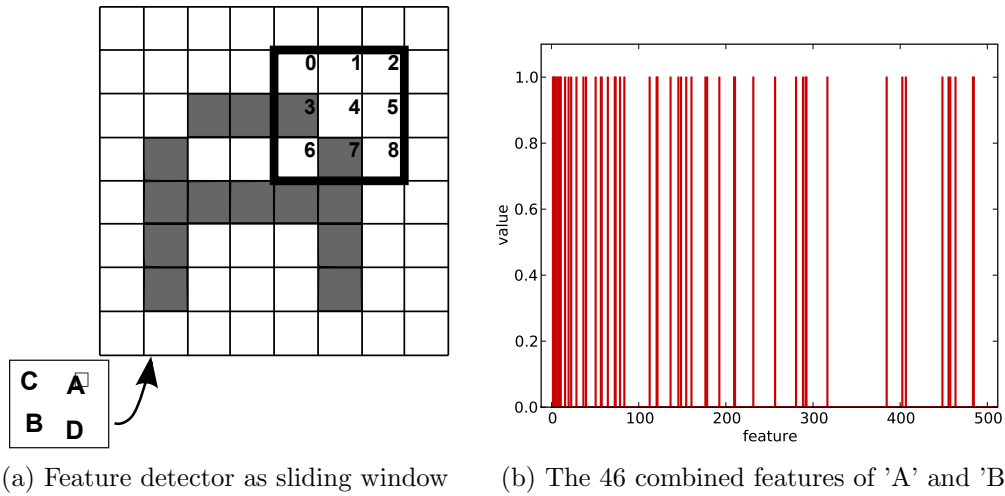


Figure 4.7: Feature detection of letter patterns. (a) Features are found by sliding a 3x3 window across the visual field, and noting all different patterns that occur within this field. Here, the window is at the top right part of the letter A, and feature $2^3 + 2^7 = 136$ is activated. (b) A has 34 features and B has 36 features, but they share 24 features. Each pattern composition is done by adding feature fields together, but we do not count multiples of the same feature. So, even if we might have 5 repetitions of one feature, we count only one. Using a 3x3 window, we get $2^9 = 512$ potential features.

As mentioned, the systems are trained on single characters. We use the sliding window to record which features are active for each single character, and use supervised learning to teach the system. Testing is done by adding multiple characters to the visual field, and having the systems disambiguating them. Figure 4.7b shows the input vector for the simultaneous presentation of characters 'A' and 'B'. While 'A' is defined by 34 different features and 'B' by 36, they share 24 features which results in 46 different input nodes being activated for this combination.

Our system thus contains 512 input nodes, and 26 output nodes. We test our system on all possible combinations of 2 to 7 simultaneous characters. We did not go any further as 7 simultaneous characters means $\binom{26}{7} = 657800$ possible combinations, which resulted in a dataset of about 660Mb and run times around 24-30 hours. We let our system run for 20 time steps on each combination and recorded the n highest activated output nodes. If all of them were equal to the patterns that were in the visual field, the trial is deemed a success. To give a taste of the difficulty of this task, imagine writing 7 different letters on top of each other, inside the space of a 5x5 box. Out of the 657800 possible combinations of all letters being different, how many times do you think you would get all of them right? Turns out that our network is able to do this quite impressively.

We compare our Fuzzy Oscillations (FO) system to artificial Neural Networks (ANN),

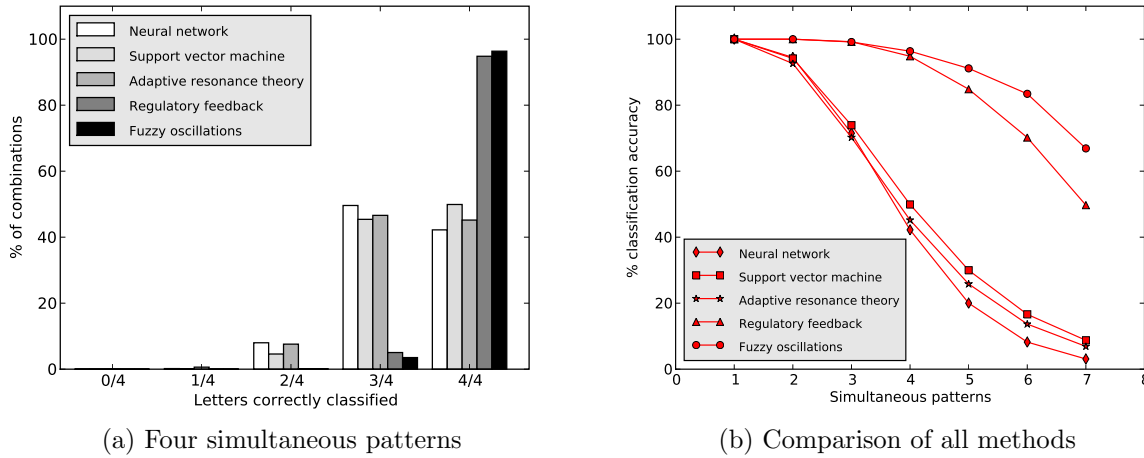


Figure 4.8: Multiple simultaneous letter patterns, where all possible combinations are tested. The top n nodes for each method is compared to the original stimuli. (a) Four simultaneously presented characters, giving $\binom{26}{4} = 14950$ possible combinations. RF and FO are much better than the other methods. Our system has 96.8% accuracy. (b) Comparison up to seven simultaneous patterns. While all methods do reasonably well with two patterns, the difference quickly gets much larger. FO outperforms RF, with 83.4% correct vs 70.2% correct at six patterns.

Support Vector Machines (SVM), Adaptive Resonance Theory (ART) and Achler's Regulatory Feedback (RF). We have touched upon all of these methods with the exception of support vector machines; they are related to ANNs, but classifies input by employing a high dimensional space and constructing hyperplanes able to separate patterns. We use the WEKA machine learning environment [56] to test ANN's, SVM's and ART while using our own implementation of RF. Each method was given single patterns to train on and tested on multiple simultaneous patterns, exactly as for our own system. However, as the WEKA environment is really designed for single category classification, we used the command-line interface to extract the normalized distributions of predictions of test instances, and handling that data in a customized script. This is equivalent to looking at the n highest activated nodes as we do on our own system.

Figure 4.8 shows the results for up to seven simultaneously presented patterns. While all methods perform satisfactory with two patterns, the four pattern scenario is substantially harder. Only the Regulatory Feedback and Fuzzy Oscillation methods are able to distinguish input patterns reasonably well. The reason for this is not some sort of incompatibility of the ANN, SVM and ART method to the problem (though they look remarkably similar in our graphs), but rather the difference between the network dynamics of those systems and the RF and FO systems. We will explain this with the use of

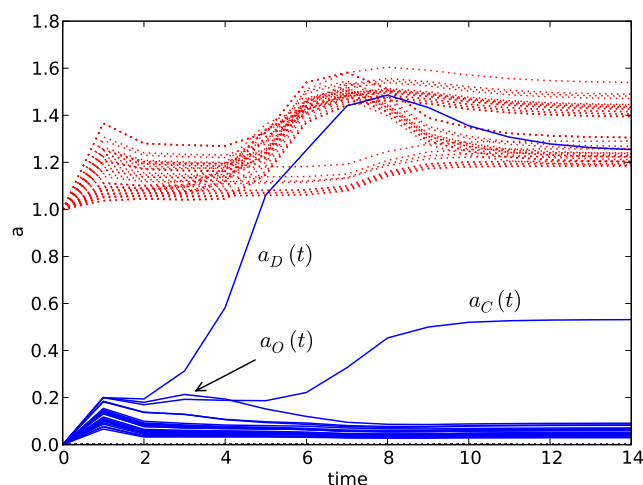


Figure 4.9: C-D example. Due to the way our feature extractor works, when the 'C' and 'D' patterns are presented simultaneously, all inputs required for the 'O' pattern is also present. In this graph, it is clear that as 'D' wins (it has more unique features than either 'O' or 'C'), the 'O' pattern loses out, thus making room for 'C'. The network stabilizes around this solution, even though the network thinks that 'O' is a part of the original stimuli at time step 3. This shows how the dynamics of the system is crucial to disambiguate multiple patterns. Red dotted lines are input nodes, blue lines are output nodes.

another example.

We show an example of two simultaneous patterns in figure 4.9. Here, our FO network is presented with the two stimuli of 'C' and 'D'. 'C' is similar to 'O' in our images: 'C' has a total of 26 features and share 24 of them with 'O'. When 'C' and 'D' is presented together, the 'D' pattern provides the rest of the features necessary to classify an 'O'. So initially, the network is unable to distinguish between 'C', 'D' and 'O'.

As the network evolves, it quickly becomes clear that 'D' is the most apparent pattern presented as it has the most unique features. As it wins, the features 'D' provided toward an 'O' is segmented to the 'D'. This results in 'O' getting less activation, and opening up the possibility for 'C'. 'C' then gains activation to the degree that the inputs shared with 'D' are redistributed away from 'D' to 'C'. This can be seen in the red dotted lines in figure 4.9. So, the important part to grasp is that the *network dynamics*, rather than any taught relationship, is what enables the network to disambiguate between patterns.

This is the major point. For the ANN, SVM or the ART to do similarly, they must be taught that explicit relationship: if 'D' and 'C' is present, the activation of 'O' should be inhibited. However, this relationship is not possible to learn when the network is taught

with single prototypical patterns! This is the core reason why ANN's, SVM's and ART performed relatively poorly above, and the situation only amplifies for more simultaneous patterns. The cases they do have right are the ones that are not ambiguous as in the 'C' and 'D' example.

The immediate implication is that for the other methods to be able to distinguish combined patterns, they must be taught those explicit relationships. This leads to a combinatorial explosion in training, where the network needs to train on any context it will encounter later. For instance, in the two-letter example above, an ANN would need to be trained for all 14950 possible combinations if it was to have any hope for 100% correct classification, which would take a prohibitively long time. A metaphor for this process is that it has to be taught that a dog can indeed be in a living room before it is able to recognize a dog in a living room. This is clearly neither feasible nor indicative of how humans learn or use our knowledge. Since both RF and FO is able to do this reasonably well, we can hypothesize that this class of algorithms, algorithms using top down feedback, are good solutions to this kind of problem.

It should of course be mentioned that ANN and SVM spent a lot of time in training, but was able to categorize very quickly. Both FO and RF spent a relatively short time in training, but comparatively longer time in categorization. This trade off however seems necessary since objects can always be encountered in unexpected circumstances.

A side note here: it does take some time for the network to settle on correct classifications, even though the trends can be available relatively early as seen in figure 4.9. Image you are out in the jungle and a tiger jumps out from a tree. You don't have time to spend to correctly classify the tiger, the trees, the bushes, and other features of the environment. So the processes we describe with our model are necessarily related to higher cognitive ability, and does not necessarily apply to the very basic fight-or-flight instincts which probably predate the neocortex. But it should be helpful that the clear trends are available early in processing.

So, given that algorithms that use top down feedback are good at separating patterns which were never encountered before, why does our system outperform regulatory feedback? As seen in figure 4.8b, at 7 simultaneous letters, RF achieves 49.7% accuracy while FO achieves 66.9%. While this being an extraordinarily difficult task, something has to account for FO being over 30% better. We suspect it is because RF does not have any concept of segmentation, only pattern separation. As seen in figure 4.9, as features are segmented to their corresponding category nodes, it becomes easier to distinguish between other patterns as they compete over the remaining features. This is a key feature of our

system, and shows the importance of segmentation in scene understanding.

To round off this section, we need to spend a few words about how we use Achler's regulatory feedback here. He does get some impressive results in [2] which we were not able to reproduce. He claims 100% correctness at 8 simultaneous patterns which are at odds with our results. However, we surmise that he does use multiple counts of each feature in his system while we only count one. This means activation to one input node could for instance be 5 while another could be only 1. We feel that by employing this mechanism, Achler artificially separates his model from others by exploiting a feature of his model. Thus, we did not use this mechanism as it disadvantages comparison with ANNs, SVMs and ART.

Chapter 5

Discussion

5.1 Model details

In many ways, the set of equations for this model is the main result of this thesis. By far most of the time spent on this project was refinement of the model itself, and we went through many revisions of it before setting on the current one. It still has potential for refinement, particularly in the distance function, but as we have seen it performs fairly well as it is.

The model converges quickly, in most cases in around 10-15 time steps. In contrast, Achler's regulatory feedback system needs in the order of 100 time steps to settle reliably for more complex datasets, and Rao et al.'s dynamical units need in excess of 200 time steps. Our model is also relatively efficient, especially in comparison to calculating each of 1000+ oscillators per population independently. But that is to be expected of the abstraction that we have done.

In our equation set, the Lorentzian was chosen as resonance function mainly for the reason that it is widely used by physicists to calculate resonance in other media. The choice of distance function was more problematic however, and we tested many different approaches to this function.

While our vector distance metric does work adequately, it does have some problems. The main problem occurs when an input node is influenced by increasingly large numbers of output nodes. Our algorithm depends on the ability for an input node to synchronize with the output node exerting the largest influence. But this gets harder as the number of output nodes increases. For example, if a node is influenced by three others where one node has twice the influence of the others, then entirely 50% of influence comes from

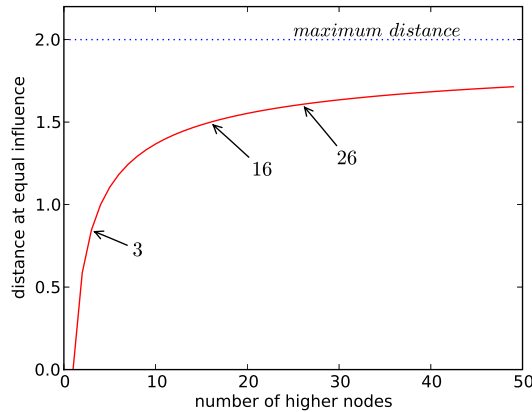


Figure 5.1: Equal influence distance as a function of connected nodes. As an input node is influenced by greater numbers of output nodes, it becomes harder to choose which node to synchronize with. Here, we show the resulting distances between an input node and n number of output nodes, where all output nodes have equal influence. As n increases, this distance approaches the maximum distance possible between nodes. The experiments performed in the previous chapter had 3, 16 and 26 output nodes respectively, which are plotted on the graph.

a single node which results in easy synchronization. However, if a node is influenced by 26 other nodes where one node has twice the influence of the others, only about 7% of influence comes from this node. So, the distance method needs to be increasingly sensitive for variations the more output nodes we have. Figure 5.1 plots the resulting distance value d when all its output nodes have equal influence, for instance at the start of a classification.

We tried a small change to our distance function as follows:

$$d_{ij}^2 = 2 \left(1 - \frac{(W_{ij}a_i)^{\frac{1}{m}}}{\sqrt{\sum_{n \in N_h} (W_{nj}a_n)^{\frac{2}{m}}}} \right) \quad (5.1)$$

where the additional m parameter has the effect of biasing the distance calculation more to the stronger inputs, thus reaching synchronization more quickly. Setting $m = 0.5$ for instance will give a mild bias to stronger inputs. We didn't use this parameter in our experiments, as we wanted to avoid extra parameters as much as possible, but this alleviated the problem to a certain degree.

This problem does limit the applicability of our method at this point, as having large networks with many possible output categories becomes more and more problematic, which can be inferred from figure 5.1. However, this might not purely be a problem

with our choice of distance function, but rather to the nature of the problem. If one node receives lots of feedback at different frequencies, it seems obvious it will have a hard time synchronizing to them. This implicitly limits the amount of higher nodes that can effectively feedback to a single lower node. To overcome this, one would probably need multiple layers of nodes and restrict the area of effect of each feedback connection. Evidently, this is exactly how the brain is organized [3]. It would be very interesting to expand our experiments to a multi-level system to attempt to capture and analyze this, but would involve the creation of rather involved experiments outside the scope of this thesis.

The initial idea of our approach was to extend the traditional neural network with the benefits of using the temporal domain. While our model does seem quite a bit different than ANNs, our model is in fact able to reduce to a normal feed-forward neural network by only changing our parameters. We can disable the effect of feedback by setting $\beta = 0$, and we can disregard oscillations at all by setting $k \gg 1$. The equations 3.9 and 3.10 then reduce to:

$$a_i = \frac{1}{N_l} \sum_{j \in N_i} W_{ij} a_j r_{ij} + I_{ext} \quad (5.2)$$

$$r_{ij} \approx 1 \quad (5.3)$$

Equation 5.3 is due to:

$$\lim_{k \rightarrow \infty} \frac{k^2}{k^2 + d^2} = 1 \quad (5.4)$$

which removes the necessity of a distance function altogether. These functions are similar to feed-forward neural networks. This allows us to bring up the distinction between feedback and recurrent networks again as mentioned in section 2.3. After removing the capability of modulating lower nodes' activity, we could still construct a recurrent neural network by taking the output of a higher node and feeding it to the input of a lower node. As we have seen however, it is the modulatory effect of feedback that is critical to solve the problems we have encountered.

There is one main difference between the above functions and artificial neural networks: the lack of any threshold function. Rolls and Treves [38], being proponents of the firing rate model, especially promote the importance of a threshold. They mention that introducing thresholds is *“an advantage, for it enables many useful computations to be*

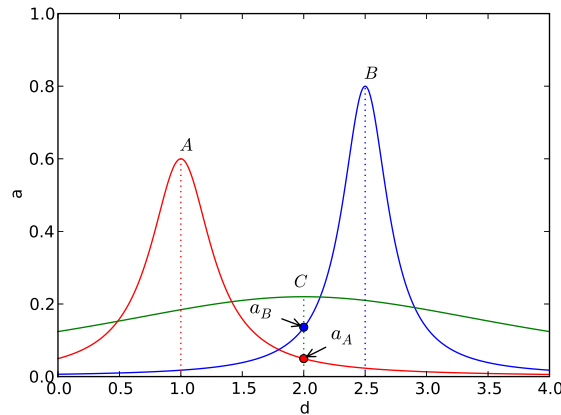


Figure 5.2: Non-linearity in activation. Here, A and B are input nodes which receive feedback influence from output node C . Activation a of C is simply the sum of a_A and a_B , weights can be assumed to be 1. As A and B move toward C , C increases activation non-linearly due to the form of the Lorentzian.

performed in neuronal networks, including removing interfering effects of similar memories, and enabling neurons to perform logical operations, such as firing only if several inputs are presented simultaneously". This begs the question, where are the thresholds in our model?

The answer is that while we do not have any threshold function *per se*, we do have high degree of non-linearity, which is the primary reason for introducing thresholds. The activity of a node is dependent on two factors: the summed activation of connected nodes and their respective degrees of synchronization. Figure 5.2 shows the activation a of an output node C . From equation 3.9, we know that the activation a of C is the sum of a_A and a_B given weights equal to 1. As A and B synchronize with C , it can be seen from the graph that a_A and a_B are non-linear due to the shape of the Lorentzian. Thus, the resulting activation of C is also non-linear. An interesting observation is that as they synchronize, a_A and a_B trace a form vaguely reminiscent of a sigmoid, the most common threshold function used in artificial neural networks.

Does this non-linearity in our case remove *"interfering effects of similar memories"* or enable *"neurons to perform logical operations"*? While we don't have exact thresholds, we have seen how we have a similarity of concept through the dynamics of the system. We haven't performed the experiments as it is outside the scope of this thesis, but we guess that some truly interesting results could be had by connecting our system to form associative and auto-associative memory systems, as described by Rolls and Treves in their 'Neural Networks and Brain Function' [38].

However, in their book they do mention one type of network called competitive networks. They are similar in setup with our networks as they are designed to categorize input vectors, but with one important distinction: competitive interaction is allowed between output neurons. This is usually implemented as lateral or mutual inhibition (for instance through a Mexican hat type function), so only a single winner emerges. This is exactly what our system also does, but without the use of any lateral connections whatsoever. The competitiveness emerges from the feedback cycle and through the dynamics of the system.

We do know, however, that there is a significant presence of lateral connections in the cortex, so how can we ignore those? Going out on a limb, we hypothesize they can be a way of inhibiting oscillations at the same frequency as neighboring populations. The effect would be to push them away so that while two neighboring populations could both be activated highly, they can not oscillate at similar frequencies. This is only an idea, but pretty neat as it would ensure that representations overlap minimally. We have not pursued this idea further in this thesis.

5.2 Learning

In our experiments, we used a simple form of Hebbian learning. If nodes oscillated together, they wired together (equation 3.12). We have previously seen the relatively short time interval required for STDP learning, in the order of ± 50 ms. We know that gamma frequency oscillations fire in the region of > 20 Hz, which means that neurons synchronized in this oscillation will be within the STDP time requirement and able to modify synaptic efficiency. We thus implicitly assume that our oscillations are within this frequency range.

However, defining link weight is a bit problematic when talking about neural populations. What exactly is a link? It is clearly not a single synapse. In section 3.5 we defined a link as a population of neurons capable of binding two populations together. The weight can thus be defined as the efficiency of this ensemble to reproduce the oscillation of the originating node. In [28], Izhikevich talks about neural groups where participation in a group is regulated by synaptic conductance delays. The modulation of synaptic strength continually changes which neurons are part of a group. He termed this process as *polychronization*. We envision a similar process being able to strengthen or weaken the link between various nodes.

A main point Izhikevich makes is that neurons can in fact be part of many different groups. We have seen this earlier, when a population could oscillate at a given frequency

while single neurons actually fire at much lower rates. This allows us to question the network structure we have used earlier. Let us elaborate.

In our experiments we have used supervised learning, where a single output neuron was forced to respond to an input pattern. We did get capacity problems when trying unsupervised learning. Now, our network structure explicitly determines the capacity by the fixed number of output nodes. Having such a fixed memory limit one would not be able to store a lifetime of learning. However, by assuming that a single neuron can indeed be a part of many different groups, it is easy to see that the same region of physical neurons could produce a potentially very large set of different populations. This is the cornerstone of Izhikevich's argument.

So, in relation to our system, we could conceivably just create additional output nodes as needed. The concept would be quite simple: have one relatively easily activated output node alongside all pattern nodes. If no output node activates more than this node, its weights are tuned to learn the pattern and another extra node is added. How easy this additional node activates would be the implicit threshold for creating new categories relative to generalizing existing categories.

In general, this method does get "dangerously" close to symbolic AI, but is plausible if we assume that individual nodes indeed can be part of many different groups. This method is also not new, it is used by Carpenter et al. in their Fuzzy ART system [9]. However, this method would require the formation of a very large number of additional output nodes, and as we saw above, our system has problems with being able to effectively distinguish between an increasingly large number of feedback connections. So before this problem is sorted out, we can not explore this idea to its potential.

Rather, our method as it stands assumes local coding at population level. This means that a category is represented by a single node. This is not very scalable. In contrast, using some sort distributed coding would greatly increase the capacity of the system. As alluded to above, using multiple layers is one way to achieve this, but more research is needed to solve this elegantly.

5.3 Connection to cognitive processes

Figure 5.3 shows a well known optical illusion: Rubin's vase. The image can either be interpreted as a vase, or two people facing each other. A special observation of this image is that one cannot see both interpretations at the same time: they are mutually exclusive.

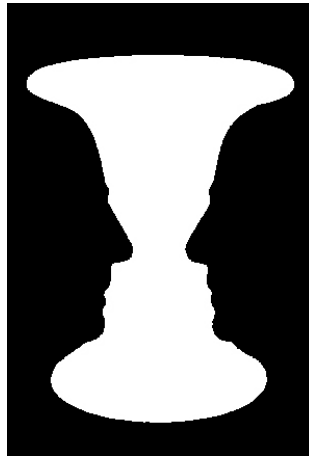


Figure 5.3: Rubin's vase: Do you see the vase or the faces?

The illusion arises due to the ambiguity in the figure-ground relationship in the image. The interpretation is based on which color one perceives as the figure (foreground) or the ground (background), and thus which shape the edges are assigned to. The Danish psychologist Edgar Rubin, who developed these illusions [40], argues that this assignment is done at a higher level than the feature recognition processes. Thus, the overall picture determines the mental interpretation rather than the net effect of each individual feature. Ambiguities in the sensed features give rise to such illusions, and these illusions are more valuable than simple party tricks: they give us a fascinating glimpse to how our brains work.

Related to what we have presented so far, this effect can be seen as an instance of perceptual segmentation. Our visual system requires the ability to divide a scene in to constituent parts to efficiently navigate and interact with the environment. This involves grouping sensed features together to represent entities, and Rubin argues with the help of the above illusion that this assignment is done top-down. As one mentally switches between interpretations, the features are segmented to different internal representations, which is the reason why it is so hard to see both interpretations at the same time in this illusion.

We surmise that if there is no ambiguity in the scene, then this can probably be solved rapidly without any need for the higher level representations to influence the lower level one. However, there is always a certain amount of ambiguity in a scene, especially when consisting of many objects partially occluding each other. Indeed, the world we inhabit and the environment we effortlessly manipulate every day is characterized by objects occluding other objects. So here we draw a parallel between the processes needed to resolve ambiguity and the processes for understanding complex scenes composed of many

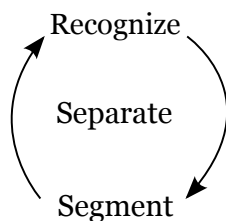


Figure 5.4: Recognition loop

objects.

We have shown how top down processing with synchronization can be used to solve this problem, where traditional approaches such as artificial neural networks perform poorly. For instance, ANNs use thresholds to determine firing rate. If an object's features are partially occluded, there is a certain chance that the correct representation will not fire at all. This is clearly not a good solution.

We expand on the analogy used earlier to visualize the difference of our system in relation to traditional artificial neural networks. When teaching concepts such as a 'dog' from picture books, young children learn to generalize that concept; it doesn't just apply to the exact image in the book, but also when seen in other contexts such as in a living room or in the back yard. The child learns to distinguish the pattern of 'dog' independently of other objects in the visual scene. From our experiment in 4.3, we have seen how feed-forward artificial neural networks need to be trained in all possible contexts to categorize such patterns, leading to a massive combinatorial explosion in training and clearly not how humans perform this task. By being able to separate and segment images as they are recognized in a scene, we can avoid this explosion.

An abstraction of the process can be depicted as in figure 5.4. Higher level recognition causes lower level segmentation, which again aids the system in recognizing objects in the scene. Through the dynamics of our model, this process of *incremental refinement* is able to separate different entities. This is at odds with current machine vision techniques which work in a primarily feed-forward manner. There, segmentation is a separate step and usually done without any domain knowledge, and usually works by grouping pixels together by color values or other similar measures (see [44]). We suspect that some good results can be had by closer integration of recognition results and segmentation in this domain.

This process of incremental refinement can be used to explain some other phenomena in visual systems. In a very good recent paper [50], Tsao and Livingstone argue that *detection* and *identification* are two distinct processes, at least for face perception. They

claim that face identification has a preprocessing stage for detecting faces, which gates the information to a specialized module for identification. The detection stage works on holistic information and detects a face by e.g. the presence of two eyes, a nose and a mouth. The identification works on more detailed information such as distances between eyes, width of nose and so on. Their main argument is that face detection and identification has two opposing demands: detection requires extracting what is common to all faces, while identification requires examination of what is different between all faces.

While this certainly seems plausible, we can now attempt to explain this differently through the process of incremental refinement. At the start of a stimulus, the activated features are not segmented to any interpretation of the scene unless some prior context gives bias toward a representation. Initially, all categories that respond to the activated features are activated. For a face this would mean that the collection of face categories would win over other potential representations, segmenting the features toward the collective group of face representation. So this would be equivalent to the detection stage of Tsao and Livingstone. As the network evolves, the features would get segmented toward specific face representations, and the network converges on a single representation, hopefully that of the correct person. This would be the identification stage. Some additional refinement is probably necessary such as using collections of features, but this could create a pretty good experiment.

We do not disagree with Tsao and Livingstone's argument about the necessity of a specialized region for face detection. For humans to be successful in social environments, it is probably vital to quickly establish a person's identity, mood and direction of attention. However, we hypothesize that such a region could have evolved from the more general process in the rest of the cortex, and thus is not necessarily an entirely algorithmically different region.

We haven't done this comparison in this paper however, as we have been more concerned with creating a general system which can be applied to multiple problem domains. We envision our system applicable toward general cognition, with feature detectors specialized for each type of sense, such as sound and touch. That is why we haven't gone deeper to visual systems and tackled problems such as invariancy. Invariancy means being able to recognize an object independent of placement (translation), size (scale) and angle (rotation). It is a hard problem, and worthy of a thesis of its own.

Chapter 6

Conclusion and future work

The main result of this thesis has been the equation set 3.9, 3.10, and 3.11, which taken together form our model of *Fuzzy Oscillations*. The relative simplicity of these equations belie the effort taken to arrive at them, as their value is in the actual dynamics of multiple connected nodes governed by them. Indeed, our goal was to develop a model capable of separation and segmentation of patterns, and this was achieved by the use of these dynamics.

The temporal binding hypothesis was developed about 10 years ago, and since then multiple authors have tackled this problem from a computational point of view. Most of these existing models for oscillatory neural networks derive from the Wilson-Cowan model or other similar oscillatory models, which are rather complex and computationally expensive. We studied Rao et al.'s dynamical units as they are representative of these approaches, but even though they use a simpler model they admit they run their system on a cluster of 24 processors. In contrast, our model easily ran on a single laptop, and thus our secondary goal of constructing an *efficient* model was achieved.

In our model, the basic unit of computation is interpreted as a set of oscillators while the above models have single oscillators. This key difference is what allows us the added efficiency and flexibility. This abstraction is admittedly a departure from neuroscience, and thus its biological plausibility can be questioned. However, we feel that this is no problem at all. We have no aspirations of becoming neuroscientists and can afford to abstract models from that domain to create useful applications in the computer science field. Indeed, our introductory discussion was about studying the brain's solution to common problems and through that inspire advancements in other fields and possibly new applications. This is exactly what we have done in this thesis.

By studying the brain’s solution to segmentation, we have developed a model capable of efficiently classifying input patterns consisting of combinations of previously learned patterns, thus effectively correctly classifying input patterns quite different than it has been exposed to before. As seen in section 4.3, this is a departure from existing classification models such as feed-forward ANNs which would rather classify a compositional pattern as a single new or unknown pattern.

We have previously mentioned that for ANNs to solve this problem, it would need to be trained on all possible combinations of objects, which would result in a combinatorial explosion in training cases. In contrast, it is the dynamics of feedback modulation in our model that enables, in a sense, the problem to be solved run-time rather than during training.

We have discussed how lower-level segmentation requires higher-level recognition and how feedback modulation is the key for this process to work. At one level of abstraction, one can say that as more and more “evidence” is gathered toward the representation of an object in a scene and its features are reserved toward this interpretation, it becomes easier to both recognize and identify other objects. This model of incremental refinement serves to reduce ambiguity in input patterns and thus make sense of the environment.

So, the following bullets describe the main points of our thesis:

- We expanded the ANN model to include temporal information in the form of synchronized oscillatory behavior.
- The synchronized oscillatory behavior coupled with differentiation between feed-forward and feedback activity enabled separation and segmentation in a competitive manner without using explicit lateral inhibitory connections.
- The resulting dynamics of the model evaded the combinatorial explosion of training by a method of continuing reduction of ambiguity.

The experiments we used in this thesis are the very first initial explorations of the capabilities of our model, and we have followed the mainstream of using visual perception as inspiration for our scenarios. However, we have attempted to keep our model as general as possible to be able to extend to other problem domains than visual perception. Our first planned extension is to add capabilities for continuous streams of input rather than static input patterns, to enable processing of primarily temporal information such as audio and video. The ultimate goal is to embody the system in some mobile agent and embed it in some interesting environment. As our model is designed to be able to recognize

complex compositional environments, we have high hopes that our approach will produce good results in the long term.

We initially wanted to include some exploratory steps in this direction in this thesis, but wisely cut out this scope creep as many other facets of our model needed to be flushed out first, primarily our problems with unsupervised learning. However, we feel our model is well suited to this sort of on-line system. The idea is that by introducing an element of inertia in the activation function, previously activated output nodes could remain active with some sort of decay time. The effect of this would be to introduce a short-term memory to the system, and could probably lead to some truly fascinating experiments.

Finally, our model was conceived as an extension to the existing ANN model, rather than by employing the above oscillator systems. By doing this, we hope to capitalize on the progress done in ANNs and apply them to our model. For example, Rolls and Treves [38] describe how they create associator, auto-associator and invariant pattern recognition networks using ANNs. It will be very interesting to apply these structures to our model and explore any new capabilities.

Over the last two decades, a huge number of artificial neural networks have been constructed to solve a huge number of different problems, including the problems we have studied in this thesis with emphasis on pattern separation. But only with the advent of oscillatory neural networks has it been possible to solve the segmentation problem. It has not been our intention to disqualify ANNs from these problems, but we have put forward the argument that the traditional ANN model needs to be extended to solve these problems as we have defined here. This extension is typically done with oscillatory mechanisms, which have proved successful in pattern segmentation. It is also not our intention to promote our model as the champion of oscillatory neural networks, as we have not performed a thorough analysis which is probably impossible as each model has it's own strengths and weaknesses.

It has been our intention, however, to develop an efficient model that exhibits the importance of feedback modulation, and show how the dynamics of the model can solve problems in quite a different manner than the neural networks dating back to Rosenblatt. Additionally, our model does not put any limitations on the type of patterns it can handle. While our model also has its shortcomings, perhaps most notably our distance function being more complex than it ideally should be, we believe we have succeeded in our vision. Most importantly, we have identified multiple paths of potential research which we will pursue in the near future.

Appendix A

Example code

The following listing is a short Python code snippet that runs the scenario in section 4.1.2 and the output is the graph in figure 4.2b. This code is in a slightly compressed form and is not really representative of the code we used in our experiments which is more object-oriented. It is presented here to give a feel for a possible implementation of our equations.

```
 -*- coding: ISO-8859-1 -*-

import math, pylab # matplotlib is required for plotting

# network
bottom_nodes = range(3) # vector of input nodes
top_nodes = range(3,6) # vector of output nodes
total_nodes = len(bottom_nodes) + len(top_nodes)

# parameters
k_b = 0.5 # k parameter for bottom level
k_t = 0.1 # k parameter for top level
beta = 0.0 # beta value regulating degree of resonant feedback
timesteps = range(10) # number of timesteps to run

# variables
a = [0.0] * total_nodes # activation level of all nodes
d = [[1.0] * total_nodes for n in bottom_nodes] # distances

# training - just set weights for now
w = [[0,0,0,1,1,0], [0,0,0,0,1,1], [0,0,0,0,0,1]] # weights

# all input nodes are activated
i = [1.0] * len(bottom_nodes) + [0.0] * len(top_nodes) # external input

# keep activity evolution for plotting
history = []
```

```

# run for a number of timesteps
for t in timesteps:

    for n_t in top_nodes: # top level
        res = sum(k_b**2 / (k_b**2 + a[n_b]**2 * d[n_b][n_t]) # resonance
                  * w[n_b][n_t] * a[n_b] for n_b in bottom_nodes)
        norm = sum(w[n_b][n_t] for n_b in bottom_nodes)
        a[n_t] = res/norm + i[n_t]

    for n_b in bottom_nodes: # bottom level
        res = sum(k_t**2 / (k_t**2 + a[n_t]**2 * d[n_b][n_t])
                  * w[n_b][n_t] * a[n_t] for n_t in top_nodes)
        a[n_b] = i[n_b]
        if i[n_b]>0.0: a[n_b] += beta * res

    for n_b in bottom_nodes: # distance function
        norm = math.sqrt(sum((w[n_b][n_t] * a[n_t])**2 for n_t in top_nodes))
        if norm>0.0: # avoid div/0 as output nodes are initialized to 0.0
            for n_t in top_nodes:
                d[n_b][n_t] = 2 * (1.0 - w[n_b][n_t] * a[n_t] / norm)

    history.append([x for x in a])

# plot using matplotlib
for i in range(total_nodes):
    y = map(lambda x: x[i], history)
    pylab.plot(timesteps, y, ['r:', 'r-'][i>=len(bottom_nodes)])
pylab.show()

```

Bibliography

- [1] M. Abeles. *Corticonics: Neural Circuits of the Cerebral Cortex*. Cambridge University Press, 1991.
- [2] T. Achler. Using Non-Oscillatory dynamics to disambiguate simultaneous patterns. *Proceedings of International Joint Conference on Neural Networks (IJCNN) 2009*, pages 41–48, 2009.
- [3] A. Angelucci and J. Bullier. Reaching beyond the classical receptive field of v1 neurons: horizontal or feedback axons? *Journal of Physiology-Paris*, 97(2-3):141–154, 2003.
- [4] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality. *Physical review A*, 38(1):364–374, 1988.
- [5] H. Berger. Über das elektrenkephalogramm des menschen. *European Archives of Psychiatry and Clinical Neuroscience*, 87(1):527–570, 1929.
- [6] G. Buzsaki. *Rhythms of the Brain*. Oxford University Press, USA, 2006.
- [7] G. Buzsaki and A. Draguhn. Neuronal oscillations in cortical networks. *Science*, 304(5679):1926–1929, 2004.
- [8] R. Callan. *Essence of neural networks*. Prentice Hall PTR Upper Saddle River, NJ, USA, 1998.
- [9] G. Carpenter, S. Grossberg, and D. B. Rosen. Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural networks*, 4(6):759–771, 1991.
- [10] G. A. Carpenter and S. Grossberg. *Pattern recognition by self-organizing neural networks*. The MIT Press, 1991.
- [11] K. Chen, D. L. Wang, and X. Liu. Weight adaptation and oscillatory correlation for image segmentation. *IEEE Transactions on Neural Networks*, 11(5):1106–1123, 2000.
- [12] F. Crick. Function of the thalamic reticular complex: The searchlight hypothesis. *Proceedings of the National Academy of Sciences*, 81(14):4586–4590, 1984.

- [13] A. K. Engel and W. Singer. Temporal binding and the neural correlates of sensory awareness. *Trends in Cognitive Sciences*, 5(1):16–25, 2001.
- [14] W. J. Freeman. *Mass action in the nervous system*. Academic Press, 1975.
- [15] W. J. Freeman. A neurobiological theory of meaning in perception. In *Neural Networks, 2003. Proceedings of the International Joint Conference on Neural Networks*, volume 2, 2003.
- [16] W. J. Freeman. The wave packet: An action potential for the 21st century. *Journal of Integrative Neuroscience*, 2:3–30, 2003.
- [17] A. Gorchetchnikov and S. Grossberg. Space, time and learning in the hippocampus: How fine spatial and temporal scales are expanded into population codes for behavioral control. *Neural Networks*, 20(2):182–193, 2007.
- [18] P. Gärdenfors. *Conceptual spaces: The geometry of thought*. The MIT Press, 2004.
- [19] G. G. Gregoriou, S. J. Gotts, H. Zhou, and R. Desimone. High-Frequency, Long-Range coupling between prefrontal and visual cortex during attention. *Science*, 324(5931):1207, 2009.
- [20] J. Hawkins and S. Blakeslee. *On Intelligence*. Oct. 2004.
- [21] D. O. Hebb. *The organization of behavior*. New York: Wiley, 1949.
- [22] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Bulletin of Mathematical Biology*, 52(1):25–71, 1990.
- [23] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.
- [24] J. J. Hopfield and C. D. Brody. What is a moment? ”Cortical” sensory integration over a brief interval. *Proceedings of the National Academy of Sciences*, page 250483697, 2000.
- [25] J. J. Hopfield and C. D. Brody. What is a moment? transient synchrony as a collective mechanism for spatiotemporal integration. *Proceedings of the National Academy of Sciences*, page 31567098, 2001.
- [26] E. M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572, 2003.
- [27] E. M. Izhikevich. Which model to use for cortical spiking neurons? *Neural Networks, IEEE Transactions on*, 15(5):1063–1070, 2004.
- [28] E. M. Izhikevich. Polychronization: computation with spikes. *Neural Computing*, 18(2):245–82, 2006.

- [29] E. M. Izhikevich. Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cereb. Cortex*, 17(10):2443–2452, Oct. 2007.
- [30] E. M. Izhikevich and G. M. Edelman. Large-scale model of mammalian thalamocortical systems. *Proceedings of the national academy of sciences*, 105(9):3593, 2008.
- [31] A. Lazar, G. Pipa, and J. Triesch. Fading memory and time series prediction in recurrent networks with different forms of plasticity. *Neural Networks*, 20(3):312–322, 2007.
- [32] R. S. T. Lee. A transient-chaotic autoassociative network (TCAN) based on lee oscillators. *IEEE Transactions on Neural Networks*, 15(5):1228–1243, 2004.
- [33] R. Legenstein, C. Naeger, and W. Maass. *What Can a Neuron Learn with Spike-Timing-Dependent Plasticity?*, volume 17 of *Neural Computation*. MIT Press, 2005.
- [34] V. B. Mountcastle. An organizing principle for cerebral function: the unit module and the distributed system. *The mindful brain*, pages 7–50, 1978.
- [35] H. Paugam-Moisy. Spiking neuron networks a survey. Survey IDIAP Research Report 06-11, IDIAP Research Institute, 2006.
- [36] R. Rao, A. Cecchi, G. A. Peck, and C. C. Kozloski. Unsupervised segmentation with dynamical units. *IEEE Transactions on Neural Networks*, 19(1):168–182, 2008.
- [37] R. Rao and G. Cecchi. An optimization approach to understanding structure-function relationships in the visual cortex. *Proceedings of International Joint Conference on Neural Networks (IJCNN) 2009*, pages 2603–2610, 2009.
- [38] E. T. Rolls and A. Treves. *Neural networks and brain function*. Oxford University Press, Oxford, UK, 1998.
- [39] F. Rosenblatt. *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Spartan Books, Washington, DC, 1962.
- [40] E. Rubin. *Synsoplevede Figurer*. Kobenhavn: Gyldendalske Boghandel, 1915.
- [41] M. N. Shadlen and J. A. Movshon. Synchrony unbound: A critical evaluation of the temporal binding hypothesis. *Neuron*, 24:67–77, 1999.
- [42] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.
- [43] S. Song, K. D. Miller, and L. F. Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3:919–926, 2000.
- [44] M. Sonka, V. Hlavac, and R. Boyle. Image processing, analysis, and machine vision second edition. *International Thomson*, 1999.
- [45] R. B. Stein. A theoretical analysis of neuronal variability. *Biophysical Journal*, 5(2):173, 1965.

- [46] S. H. Strogatz. From kuramoto to crawford: exploring the onset of synchronization in populations of coupled oscillators. *Physica D: Nonlinear Phenomena*, 143(1-4):1–20, 2000.
- [47] S. H. Strogatz. *Sync: How order emerges from chaos in the universe, nature, and daily life*. Hyperion, New York, 2003.
- [48] A. Treisman. The binding problem. *Current Opinion in Neurobiology*, 6(2):171–178, 1996.
- [49] A. Treisman and S. Gormican. Feature analysis in early vision: Evidence from search asymmetries. *Psychological Review*, 95(1):15–48, 1988.
- [50] D. Y. Tsao and M. S. Livingstone. Mechanisms of face perception. *Annual Review of Neuroscience*, (31):411–437, 2008.
- [51] C. von der Malsburg. The correlation theory of brain function. *Models of Neural Networks II: Temporal Aspects of Coding and Information Processing in Biological Systems*, pages 95–119, 1994.
- [52] C. von der Malsburg and J. Buhmann. Sensory segmentation with coupled neural oscillators. *Biological Cybernetics*, 67(3):233–242, 1992.
- [53] D. L. Wang and X. Liu. Scene analysis by integrating primitive segmentation and associative memory. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 32(3):254–268, 2002.
- [54] H. R. Wilson. *Spikes, Decisions, and Actions: The Dynamical Foundations of Neuroscience*. Oxford University Press, USA, 1999.
- [55] H. R. Wilson and J. D. Cowan. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical Journal*, 12(1):1, 1972.
- [56] I. H. Witten and E. Frank. Data mining: practical machine learning tools and techniques with java implementations. *ACM SIGMOD Record*, 31(1):76–77, 2002.