Nada är en gemensam institution mellan
Kungliga Tekniska högskolan och Stockholms universitet.

An Attractor Memory Model of Neocortex

CHRISTOPHER JOHANSSON

Avhandling som med tillstånd av Kungliga Tekniska högskolan
framlägges till offentlig granskning för avläggande av teknologie doktorsexamen
torsdagen den 26 oktober 2006 kl 10.15
i sal F2, Lindstedtsvägen 28,
Kungliga Tekniska högskolan, Stockholm.

ISBN 91-7178-461-6
TRITA-CSC-A-2006:14
ISSN-1653-5723
ISRN-KTH/CSC/A--06/14--SE

Abstract

This thesis presents an abstract model of the mammalian neocortex. The model was constructed by taking a top-down view on the cortex, where it is assumed that cortex to a first approximation works as a system with attractor dynamics. The model deals with the processing of static inputs from the perspectives of biological mapping, algorithmic, and physical implementation, but it does not consider the temporal aspects of these inputs. The purpose of the model is twofold: Firstly, it is an abstract model of the cortex and as such it can be used to evaluate hypotheses about cortical function and structure. Secondly, it forms the basis of a general information processing system that may be implemented in computers. The characteristics of this model are studied both analytically and by simulation experiments, and we also discuss its parallel implementation on cluster computers as well as in digital hardware.

The basic design of the model is based on a thorough literature study of the mammalian cortex's anatomy and physiology. We review both the layered and columnar structure of cortex and also the long- and short-range connectivity between neurons. Characteristics of cortex that defines its computational complexity such as the time-scales of cellular processes that transport ions in and out of neurons and give rise to electric signals are also investigated. In particular we study the size of cortex in terms of neuron and synapse numbers in five mammals; mouse, rat, cat, macaque, and human. The cortical model is implemented with a connectionist type of network where the functional units correspond to cortical minicolumns and these are in turn grouped into hypercolumn modules. The learning-rules used in the model are local in space and time, which make them biologically plausible and also allows for efficient parallel implementation. We study the implemented model both as a single- and multi-layered network. Instances of the model with sizes up to that of a rat-cortex equivalent are implemented and run on cluster computers in 23% of real time. We demonstrate on tasks involving image-data that the cortical model can be used for meaningful computations such as noise reduction, pattern completion, prototype extraction, hierarchical clustering, classification, and content addressable memory, and we show that also the largest cortex equivalent instances of the model can perform these types of computations. Important characteristics of the model are that it is insensitive to limited errors in the computational hardware and noise in the input data. Furthermore, it can learn from examples and is self-organizing to some extent. The proposed model contributes to the quest of understanding the cortex and it is also a first step towards a brain-inspired computing system that can be implemented in the molecular scale computers of tomorrow.

The main contributions of this thesis are: (i) A review of the size, modularization, and computational structure of the mammalian neocortex. (ii) An abstract generic connectionist network model of the mammalian cortex. (iii) A framework for a brain-inspired self-organizing information processing system. (iv) Theoretical work on the properties of the model when used as an autoassociative memory. (v) Theoretical insights on the anatomy and physiology of the cortex. (vi) Efficient implementation techniques and simulations of cortical sized instances. (vii) A fixed-point arithmetic implementation of the model that can be used in digital hardware.

Keywords: Attractor Neural Networks, Cerebral Cortex, Neocortex, Brain Like Computing, Hypercolumns, Minicolumns, BCPNN, Parallel Computers, Autoassociative Memory

ISBN 91-7178-461-6 • TRITA-CSC-A-2006:14 • ISSN-1653-5723 • ISRN-KTH/CSC/A--06/14--SE

Acknowledgements

First of all, I want to express my utmost gratitude to my primary supervisor professor Anders Lansner for his support and enthusiasm, and also to my secondary supervisor Dr. Örjan Ekeberg for insightful comments and enlightening discussions. I also want to thank Dr. Anders Sandberg, Mikeal Djurfeldt, Peter Raicevic, and Martin Rehn for productive collaborations and inspiring discussions. I am very grateful for the many invaluable advises I have received from Dr. Erik Fransén and Dr. Jeanette Hellgren Kotaleski. Great thanks to all former and present members of the SANS research group for a nice and pleasant atmosphere and lively discussions. I am very grateful for the support and encouragement I have received from Sofia and my family.

This work was partly supported by grants from the Swedish Science Council (Vetenskapsrådet) grant No. 621-2001-2548, 621-2003-6256, 621-2004-3807, and from the European Union (FACETS project, FP6-2004-IST-FETPI-015879).

Contents

1 Introduction

The purpose of this work is twofold: Firstly, to build a functional model of the mammalian neocortex. Secondly, to develop a new massively parallel and self-organizing information processing system that builds on the data-processing principles employed by the brain, in particular the neocortex.

The largest component of the mammalian brain is the cerebral cortex and in humans more than 95% of it is a six-layered structure referred to as neocortex. Neocortex is characterized by having a very homogenous arrangement of its micro-circuitry and a remarkable feature of it is that it processes inputs from several different modalities. Because of cortex's homogenous structure it is tempting to speculate that it implements some form of general and generic computational machinery that can be used to process a wide range of different types of data. Assuming this is the case, it would be very interesting to characterize and understand this general underlying computational structure and that is the starting point of this thesis.

Regarding the first objective we here chose to build a top-down, abstract, model of cortex. This type of model may be interesting both for cognitive psychologists (studies memory, attention, and perceptual integration) and for neuroscientists (studies the properties of single synapses, single neurons, and small ensembles of neurons). In the case of cognitive psychologists this type of model gives them a tool with which they can test, explain, and verify new hypotheses and ideas about brain function. In the case of neuroscientists this type of model is interesting because it can be used to explain global, emergent, patterns of activity in large ensembles of neurons and the architecture of cortex on a macroscopic scale. It also provides a link to more detailed, less abstract, models that in turn can be used to explain the firing patterns of individual neurons and how these neurons are affected by different ion channels. Hopefully this type of abstract functional models can help to elucidate the causes of neurological disorders and may provide clues to treatments of such malfunctions.

The second task, building a brain-inspired computing system, also requires an in-depth knowledge about the brain's physiology and function. Here we try to design new algorithms, and in the end also silicon hardware, that function on the same principles as the brain. Computer scientists have been busy the last three decades trying to solve tasks such as human like intelligence, speech recognition, object recognition, and autonomous navigation. The human brain and cortex readily perform all these tasks. This is a strong argument, from an engineering perspective, for trying to understand how the cortex works. This approach is often referred to as *brain like computing*. Three important aspects of these brain-like algorithms, in which they differ from traditional computer algorithms, are that they have the ability to learn, they adapt to the input, and they are intrinsically massively parallel. The learning is needed for the algorithms to be flexible and have the ability to adapt to new situations and conditions. Further, systems built with this type of algorithms are likely to have the ability to solve boundary problems efficiently, e.g. visual object identification and voice recognition, better than today's algorithms.

On the way to the goal of a cortical model there are many considerations to be made. The data on biological systems often have large variations and is incomplete. The functional properties of biological components are often multifaceted, and the functional dependencies are numerous. Causal relations are often hard to deduce, although in primary sensory areas, e.g. the visual cortex, they can be found. It is therefore very important, and also very hard, to find a model with an appropriate level of abstraction. If the model is too complex it is not possible to run a large-scale version of it. On the other hand, if the model is too simplistic, it may not capture the relations necessary for the functionality present in cortex to arise. Finding such a model is also difficult because of the large differences in the approach taken by biologists and computer scientists to understand the cortex. Biologists often want to consider every single detail of what they are studying while computer scientists are more prone to put things into a general framework and find out the underlying processing principles. A striking example is that when a biologist and a computer scientist are asked how many different types of neurons there are in cortex, the biologist answers ten million and the computer scientist says two (Toulouse 1989). Thus, overcoming this barrier is a key to successfully building a cortical model that has an appropriate level of abstraction and in this thesis we have tried to achieve this by thoroughly reviewing empirical data on cortex. We will assume that the functional principles of cortex reside at a level of abstraction higher than at the level of the single neuron but closer to that of a neural network of the type used in connectionist models (Feldman and Ballard 1982; Medler 1998).

A common trade among neuroscientists and cognitive psychologists is to present conceptual models of how the brain processes information. These models are often vaguely described and typically there are lots of pieces of information missing before a working functional system can be realized based on these conceptual ideas. Marr (1982) refers to these conceptual models as *computational theory*. He also distinguished two other levels at which any information processing system can be understood (Marr 1982); *representation and algorithm* and *hardware implementation*. Computational theory is the most abstract description, only describing the basic principles underlying the information processing in the system. Representation and algorithm is the detailed blue print of the system. The last level, hardware implementation, is concerned with the physical implementation of the system. In this thesis we first present a conceptual model of cortex (computational theory), then we give a detailed description of its implementation (representation and algorithm), and finally we discuss implementations of the model on both clusters and in hardware (hardware implementation).

In technical terms the design of a functional model of cortex is difficult because the problem is under constrained and both the constraints and objectives of the design are somewhat diffuse. For a cortical model there are two types of constraints. Anatomical constraints, such as the modularization and connectivity of neurons, and physiological constraints; e.g. that only a few neurons can be active at any one time and that a neuron need a fair amount of synaptic input to fire action potentials. Neither can we in precise terms define what the cortical model should be optimized for and the intuitive solution to this problem is to build a bottom-up model, modeling all ion-channels in all neurons in the entire cortex and then hope that interesting functionality will arise and pop out. Unfortunately, that approach is probably unfeasible because of the immense complexity of cortex. Instead we take a top-down view on cortical function and assume that it to a first rough approximation is that of an auto-associative memory implemented by an attractor network.

Reasoning about optimal functionality is a valuable guiding tool in the design of a functional cortical model but it has to be done with care. Using this type of engineering approach when constructing a cortical model can be very misleading if the costs and constraints that have formed and shaped cortex throughout evolution are not fully understood or known.

Why do we study the neocortex of highly evolved mammals and not the neural circuitry of a simpler animal? The reason for this is that we believe that the functional principles, similar to the anatomical and physiological properties of cortex are quite homogenous both across cortical areas and mammal species. The neocortex is the latest addition to the brain circuitry in an evolutionary perspective. Before the cortex was evolved, the functionality of the nervous systems was extended by increasing the number of nucleus (spherical group of neurons) and also by creating task specific neurons. The function of these pre-cortical circuitries is often adapted and unique to a particular species and genus. Because of cortex's late evolutionary arrival and its homogenous appearance it is reasonable to expect that it performs a general type of computation that can be applied in different forms to a wide range of tasks. Further, cortex shows an exceptional scaling with over three orders of magnitude from the smallest mammals to humans.

The inherent parallelism and good scaling properties of the cortex are likely to be present in a model of it and these properties would make the model well suited for an implementation on a parallel computational hardware. The intrinsic parallelism of this type of algorithms also makes them well suited for the next generation of hardware and computing devices that are going to be increasingly parallel. The trend towards parallel computing systems is fueled by the need to keep down power consumption and also find use for the ever more increasing number of transistors present in electronic devices. An important aspect of the computations implemented by the brain like algorithms discussed in this thesis is that they are redundant to both errors in the hardware and noisy input data. These algorithms are typically also modular, which is a big advantage in the design of a complex system.

Our vision is that we one day will be able to emulate an artificial neural system with a size and complexity comparable to the human cortex. Having the means to run brain-sized artificial neural systems would take us closer to the goal of making intelligent programs. The target computational structure in our vision is not a parallel computer that fills an entire room, but a small structure no larger than a human brain. Being able to fit brain-sized systems into small, low power consuming, electronic devices would make it possible to build truly intelligent robots and electronic agents.

In this thesis we characterize the neocortex of mammals, then propose a model of it, implement the proposed model on a parallel computational hardware, and finally we show that it can be used for meaningful computations.

1.1 Contributions

- We propose an abstract model of cortex and investigate its properties thoroughly. This model is used to implement both autoassociative memory and classifier networks, and in particular we investigate the storage capacity of the former type of network. Furthermore, we derive an energy function for an approximation of the network implementing the cortical model and investigate the properties of its memory attractors. Appendices I-VIII

- We use the model to explain some aspects of cortical anatomy and physiology. These aspects include the modular organization of cortex and the patchy layout of synaptic connections, and also the sparse activation of cortical neurons. Appendices III and VIII
- We present a framework for a brain-inspired computing system based on the model. We exemplify the computational capabilities of this framework on a number of tasks involving image data and when used as an autoassociative memory this system can store arbitrary types of patterns. Appendices I, VI, VII
- We present performance and scaling results from implementations of the model. (These instances of the model are the world's currently largest neural networks.) We investigate the communication requirements of the model and experiment with an innovative approach to efficient communication in commodity clusters with standard Ethernet interconnect. We present a fixed-point arithmetic implementation of the model and discuss its hardware implementation. Appendix I and V

1.2 Thesis Structure

In chapter 2 we give an introduction to the structure and components of the neocortex. We investigate five differently sized cortices, from mouse to man. We discuss neurons and synapses in terms of number and characteristics, together with cortical layers and cortical columns in terms of structure and size.

Based on the through review of cortical anatomy we present an abstract hierarchical model of cortex in chapter 3. Different physiological aspects of the computational capabilities of cortical neurons are also discussed.

In chapter 4 we review the field of attractor neural networks with a focus on their use as autoassociative memory.

Next, in chapter 5, a special class of attractor neural networks that have a modular structure is presented. In particular we present the Bayesian Confidence Propagating Neural Network (BCPNN), which has this modular structure. We discuss the storage capacity of the BCPNN and analyze its dynamics.

In chapter 6 we exemplify the type of computations performed by an attractor network and show how the hierarchical cortical model instantiated with a BCPNN extends the functionality of the basic single layered attractor network. A number of experiments using images are presented and we also present results on using the hierarchical network as a classifier.

In chapter 7 we first present an implementation of BCPNN on cluster computers. Secondly, we review the levels of parallelism in BCPNN and discuss the communication requirements. Thirdly, we present an efficient implementation of the communication on clusters interconnected by Ethernet.

Chapter 8 contains a discussion on different techniques for implementing neural networks in hardware. We present a fixed-point arithmetic implementation of BCPNN and discuss an implementation of BCPNN on the Mitrion co-processor hardware.

In chapter 9 we discuss the results of this thesis.

In chapter 10 we present the conclusions of this thesis and give hints for possible future extensions of the presented work.

1.3 Articles

This thesis is based on the following eight papers in appendices I-VIII. Five of these papers have been published in or submitted to international journals and three of these papers have been presented at international conferences.

I. Christopher Johansson, Anders Lansner, "Towards Cortex Sized Artificial Neural Systems", Neural Networks (in press)

II. Christopher Johansson, Anders Sandberg, Anders Lansner, "A Neural Network with Hypercolumns", 2002, in Proc. International Conference on Artificial Neural Networks - ICANN'02, Aug. 28-30, Madrid, Spain, Springer-Verlag Berlin, p.192-197, LNCS 2415

III. Christopher Johansson, Martin Rehn, Anders Lansner**,** "Attractor Neural Networks with Patchy Connectivity", 2006, Neurocomputing, Vol. 69(7-9), p. 627-633

IV. Christopher Johansson, Örjan Ekeberg, Anders Lansner, "Clustering of stored memories in an attractor network with local competition", International Journal of Neural Systems (submitted)

V. Christopher Johansson, Anders Lansner, "Implementing Plastic Weights in Neural Networks using Low Precision Arithmetic", International Journal of Neural Systems (submitted)

VI. Christopher Johansson, Anders Lansner, "Attractor Memory with Self-Organizing Input", 2006, in Proc. Biologically Inspired Approaches to Advanced Information Technology – BioADIT'06, Jan. 26-27, Osaka, Japan, p. 265-280, LNCS 3853

VII. Christopher Johansson, Anders Lansner, "A Hierarchical Brain Inspired Computing System", 2006, in Proc. International Symposium on Nonlinear Theory and its Applications – NOLTA'06, Sep. 11-14, Bologna, Italy, p. 599-602

VIII. Christopher Johansson, Anders Lansner, "Imposing Biological Constraints onto an Abstract Neocortical Attractor Network Model", Neural Computation (in press)

Other papers by the author that are related but not included in the thesis:

I. Christopher Johansson, Martin Rehn, Anders Lansner, "Attractor Neural Networks with Patchy Connectivity", 2005, in Proc. The 13th European Symposium on Artificial Neural Networks – ESANN'05, Bruges, Belgium, p. 429-434

II. Christopher Johansson, Anders Lansner, "A Cortex Sized Artificial Attractor Neural Network Applied to Image Restoration", 2005, April 12-14, SAIS-SSLS Joint Workshop, Mälardalen University, Västerås, Sweden

III. Christopher Johansson, Anders Lansner, "Towards Cortex Sized Artificial Nervous Systems", 2004, Knowledge-Based Intelligent Information and Engineering Systems – KES'04, Wellington, New Zealand, LNAI 3213, p. 959-966

IV. Christopher Johansson, Anders Lansner, "Towards Cortex Sized Attractor ANN", 2004, in Proc. Biologically Inspired Approaches to Advanced Information Technology – BioADIT'04, Lausanne, Switzerland, LNCS 3141, p. 63-79

V. Christopher Johansson, Peter Raicevic, Anders Lansner, "Reinforcement Learning Based on a Bayesian Confidence Propagating Neural Network", 2003, April 10-11, SAIS-SSLS Joint Workshop, Center for Applied Autonomous Sensor Systems, Örebro, Sweden

VI. Mikeal Djurfeldt, Christopher Johansson, Örjan Ekeberg, Martin Rehn, Mikeal Lundqvist, Anders Lansner, “Massively parallel simulation of brain-scale neuronal network models”, 2005, TRITA-NA-P0513, Department of Numerical Analysis and Computer Science, Royal Institute of Technology

VII. Christopher Johansson, Anders Lansner, "A Mean Field Approximation of BCPNN", 2005, TRITA-NA-P0506, Royal Institute of Technology, Department of Numerical Analysis and Computer Science

VIII. Christopher Johansson, Anders Lansner, "On the Storage Capacity of an Abstract Cortical Model with Silent Hypercolumns", 2005, TRITA-NA-P0501, Department of Numerical Analysis and Computer Science, Royal Institute of Technology

IX. Christopher Johansson, Anders Lansner, "BCPNN Implemented with Fixed-Point Arithmetic", 2004, TRITA-NA-P0403, Department of Numerical Analysis and Computer Science, Royal Institute of Technology

X. Christopher Johansson, Anders Lansner, "Mapping of the BCPNN onto Cluster Computers", 2003, TRITA-NA-P0305, Royal Institute of Technology, Department of Numerical Analysis and Computer Science

XI. Christopher Johansson, Anders Lansner, "A Neural Reinforcement Learning System", 2002, TRITA-NA-P0215, Department of Numerical Analysis and Computer Science, Royal Institute of Technology

XII. Christopher Johansson, Anders Lansner, "A Parallel Implementation of a Bayesian Neural Network with Hypercolumns", 2001, TRITA-NA-P0121, Department of Numerical Analysis and Computer Science, Royal Institute of Technology

XIII. Christopher Johansson, Anders Sandberg, Anders Lansner, "A Capacity Study of a Bayesian Neural Network with Hypercolumns", 2001, TRITA-NA-P0120, Department of Numerical Analysis and Computer Science, Royal Institute of Technology

2 The Cerebral Cortex of Mammals

In this chapter we give a quick introduction to the nervous system of mammals. We then turn to study the cerebral cortex of mammals. We study the structure of cortex in terms of areas, layers, and columns. The building blocks of cortex are also studied and quantified, i.e. the neurons and their synapses. We study how the neurons are interconnected, both on a microscopic and macroscopic scale.

This chapter primarily discusses the content of appendix I but it also presents some of the data used in appendices III, VI, VII, and VIII.

2.1 An Overview of the Mammalian Nervous System

All higher life forms on this planet have a nervous system that receives and processes information. A nervous system is built from neurons (nerve cells). Neurons differ from other cells in that they have specialized in regulating and conveying their electric potential. It is not only neurons that work with electrical potentials; muscle, endocrine, and sensory cells also make use of electrical potentials. A typical neuron has multiple dendrites (a dendritic tree), a soma (cell body), and an axon (Figure 2.1). Neurons communicate with electric impulses that are conveyed by the their axon and these impulses are initiated in the axon hillock. These electrical impulses are received mostly via chemical synapses in the dendritic tree of the postsynaptic neurons, but they can also be received by synapses located on the axon and soma. In a chemical synapse the signal is conveyed by the release of a neurotransmitter in the synaptic cleft, i.e. the space at the crossing between a presynaptic axon and a part of the postsynaptic neuron. Depending on the type of neurotransmitter either an excitatory or an inhibitory postsynaptic potential is generated (EPSP or IPSP). The strength of the postsynaptic potential depends mainly on two variables, the activity of the presynaptic neurons and the efficiency of the synapses. In addition chemical synapses, neurons can also communicate via gap junctions that passively conveys a signal and these are also referred to as electrical synapses.

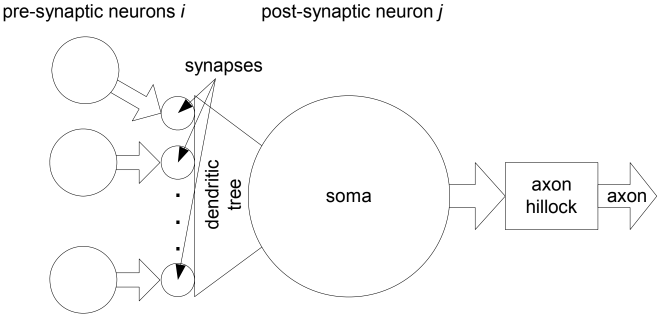


Figure 2.1. A schematic model of a neuron.

Coupled together in huge numbers (approximately 10^{11} in humans) the neurons form the central nervous system (CNS). The major components of the CNS are the spinal cord, brain stem, cerebellum, and cerebral cortex. The spinal cord relay motor signals from the brain and it also incorporates circuits for autonomous and reactive control. The brain stem is the center for many vital, life supportive, functions such as respiration. Cerebellum helps the cerebral cortex in processing motor commands but it also contributes to other cognitive functions requiring a precise timing. In the cerebral cortex all complex, higher order, cognitive functions are located and it can be said to be the command and control central of the CNS (Gazzaniga, Ivry et al. 1998).

In adult humans the brain accounts for approximately 2% of the body's weight but it stands for roughly 20% of the energy consumption (Kandel, Schwartz et al. 2000; Attwell and Laughlin 2001). Given that the body of an active human male consumes energy at a rate of 150 W (Henry 2005) the brain uses 30 W. Thus, the brain is very expensive in terms of energy, on parity with a light bulb, but apparently this expense pays off in an evolutionary perspective.

2.2 An Overview of the Cerebral Cortex

The brain nomenclature is extensive (Heimer 1988). There are several different divisions of the brain based on developmental, functional, and anatomical aspects. A single component of the brain can often have multiple names depending on the type of division used and species. In the following we give a short roundup of the most common divisions and terminology.

Based on appearance the cerebral cortex is divided into isocortex with a homotypical structure and allocortex that has a heterotypical structre. Isocortex is also referred to as neocortex. Allocortex is composed of the olfactory (piriform) cortex and the hippocampal structures. A division of the cerebral cortex based on phylogenetic differences states that it is composed of three parts (Northcutt and Kaas 1995); the neo-, paleo-, and archicortex (hippocampus). The paleocortex is evolutionarily older than the neocortex and it is mainly composed of the olfactory cortex and the parahippocampal gyrus i.e. structures right next to hippocampus. In the literature a distinction between cerebral cortex and neocortex is not always made, and these structures are sometimes referred to simply as cortex (DeFelipe and Fariñas 1992). In humans, roguhly 95% of the cerebral cortex is neocortex and in other mammals it is somewhat less. The term cortex means "bark" as in tree bark.

The cerebral cortex encompasses a volume called cerebrum. The cerebrum is physically divided into two symmetrical hemispheres that are connected via the corpus callosum. Furthermore, it is physically divided into four lobes; the frontal, parietal, temporal, and occipital lobe.

Based on functional and anatomical properties, cortex is dived into areas. The best known division is that of Broadmann. The frontal areas of cortex are attributed higher cognitive processes such as abstract thinking and planning. The prefrontal and parietal (central) areas of cortex are devoted to motor command and somatosensory input from the body. The areas located in the temporal lobes are devoted to auditory processing. In the back of cortex, the striate cortex (striate cortex, V1, area 17, and primary visual cortex are all synonyms) is located, which is devoted to visual processing. Areas that receives input from more than one modality (sensory system) is referred to as multimodal or association cortex. Associative capabilities exist in all cortical areas.

The cortex of mammals is generally quite homogenous (Rockel, Hiorns et al. 1980) and it has seen a great increase in size during evolution. In humans the cortex is ~3 mm thick (Hofman 1985; Nieuwenhuys, Donkelaar et al. 1997) and in mouse ~0.8 mm (section 2.8.3).

The differences in cortical thickness between small and large mammals is attributed the increased need of neuropil for wiring in larger mammals (Karbowski 2003).

In large mammals the cortex is folded in order to save space and pack a larger cortical area into the skull. In smaller mammals such as mouse the cortex is not folded. When the brain volume is larger than ~4 cm^2 (mammals with the size of rats and up) the cortex starts to fold (Hofman 1988).

In mammals the cerebrum constitutes a major part of the brain. In small mammals ~60% of the brain is occupied by the cerebrum and ~75% in primates (Clark, Mitra et al. 2001). In turn, the neocortex occupies ~55% of the cerebral cortex in granivores (e.q. squirrel and mouse), ~80% in primates, and ~95% in humans (Clark, Mitra et al. 2001).

An interesting property that more or less applies to all mammals is a constant neuron density in cortex of $\sim 10^5$ mm^{-2} (Rockel, Hiorns et al. 1980; Braitenberg and Schuz 1998) except for V1 in primates where it is ~2.5 times higher (Rockel, Hiorns et al. 1980; Beaulieu, Kisvarday et al. 1992). But there are reports that state that the neuron density per area is smaller in larger mammals (Haug 1987).

2.3 Neurons and Synapses in the Cerebral Cortex

The proportions between different types of neurons are fairly constant between different areas (Rockel, Hiorns et al. 1980) and mammalian species (Beaulieu, Kisvarday et al. 1992).

Different types of neurons have different numbers of synapses and typically it varies between $2\cdot 10^3$ and $2\cdot 10^4$ (Braitenberg and Schuz 1998), but there are cortical neurons that have as many as $6\cdot 10^4$ synapses (DeFelipe and Fariñas 1992). The average number of synapses per neuron (for all types) in mouse (Braitenberg and Schuz 1998), cat (Beaulieu and Colonnier 1989), and man (Pakkenberg, Pelvig et al. 2003) is $\sim 8\cdot 10^3$, but this figure can vary a factor 2-3 (section 2.8.5).

In the cerebral cortex of mammals there are two basic types of cells; glia and nerve cells. The role of glia cells is to support nerve cells, e.g. by providing isolating myelin for the axons. Neurons come in many different forms and they are often classified based on the morphology of their dendrites and axons as; unipolar, bipolar, multipolar, and psuedopolar. In cortex the most common type of morphology is the multipolar.

Approximately 75-80% of the neurons in cortex are excitatory pyramidal neurons and the remaining 10-25% are inhibitory interneurons (Braitenberg and Schuz 1998; Buxhoeveden and Casanova 2002). There are also a few types of non-pyramidal excitatory neurons e.g. stellate neurons that have a spherical, star shaped, dendritic tree. Neurons such as the stellate cells, that have localized and concentrated dendritic and axon arborizations, are called interneurons.

Cortical neurons are usually divided into those that have spines or not (DeFelipe and Fariñas 1992). Pyramidal and spiny stellate neurons, amongst others, have spines. Common spine-less neurons are the basket, chandelier, and double bouquet neurons. Neurons with spines are generally excitatory, while spine-less neurons typically are inhibitory. Approximately 70-80% of all neurons have spines while 20-30% do not (DeFelipe and Fariñas 1992).

The inhibitory interneurons are divided into two classes depending on their axonal targets (Thomson and Bannister 2003). Two types of interneurons that target parts of the pyramidal neurons close to the soma and locations on the axon are the chandelier and basket neurons. Examples of interneurons that target the dendrites of pyramidal neurons are the double bouquet, Martinotti, and spider web neurons. Basket neurons come in many different sizes and shapes, but they are generally divided into large and small basket neurons. Large basket neurons are the largest of all inhibitory neurons. Basket neurons are present in all layers, but the large basket neurons are most common on the border between layer III and IV (Thomson and Bannister 2003; Markram, Toledo-Rodriguez et al. 2004). Double bouquet neurons are bipolar and have concentrated axon bundles that are both ascending and descending. They exist in layers II-IV (DeFelipe, Hendry et al. 1990; Thomson and Bannister 2003).

Excitatory and inhibitory synapses are commonly denoted as type I and type II synapses or as asymmetrical and symmetrical synapses (DeFelipe and Fariñas 1992). 75-95% of all synapses in the human cortex are excitatory while 5-25% are inhibitory (DeFelipe, Marco et al. 1999). GABA is the major inhibitory neurotransmitter (Beaulieu, Kisvarday et al. 1992) and glutamate is the most common excitatory neurotransmitter in cortex (Kawaguchi and Kubota 1997). Approximately 95% of the synapses on a neuron are located in the dendritic tree (DeFelipe and Fariñas 1992).

The resting potential of a neuron is normally between −40 and −70 mV. The resting potential is a result of active transportation of ions, mainly Na^+, Ca^{2+}, Cl^-, and K^+, through ion-channels in the cell membrane. It has been estimated that there are a couple of hundred different types of ion-channels in all types of living cells. In humans there are at least fifty different types of ion-channels (Hille 1992).

2.4 Cortical Layers

Neocortex is divided into six layers based on a morphologic division[1] and this distinguishes it from allocortex that has less than six layers. Here we attempt to give a general overview of the layered architecture of neocortex and its functionality. The exact structure and size of the different layers differ between different cortical areas and mammalian species.

When one state that a neuron belongs to a certain layer, it means that its soma is located in this layer. The main part of the dendritic arborization of a neuron may very well be located in a different layer than its soma. The same is true for the boutons, i.e. the terminations of the axons.

2.4.1 The Layers of Neocortex

Layer I is the most superficial layer and it occupies approximately 10% of the cortical depth. This layer is dominated by dendrites and axons of neurons in the lower layers as well as a few inhibitory neurons; ~85% of the neurons are inhibitory (Beaulieu, Kisvarday et al. 1992). This layer also has a large number of horizontal fibers running a few mm (Eccles 1981). Afferent axons of layer V pyramidal neurons terminate in this layer (Douglas and Martin 2004).

Layers II and III are often taken together and referred to as layers II/III. Layers II/III is mainly populated by pyramidal neurons. The smallest pyramidal neurons are located in the superficial part of layer II. The size of the pyramidal neurons increases continuously as one moves down to layer III. Pyramidal neurons in this layer locally target other pyramidal neurons in layer V and interneurons in layer IV (Thomson and Bannister 2003; Douglas and Martin 2004). The main bulk of axons, about 75%, of layer III pyramidal neurons are long and stretch out horizontally and connects with other layer III neurons (Martin 2002).

---

[1] See Peters and Sethares (1991) for an illustration of the cortical layers.

Layer IV is seen as the input layer of cortex. In cortical areas devoted to sensory input layer IV is often quite thick. Afferents from other cortical areas and deep brain structures (mainly from the thalamic region) terminate in this layer. Neurons in layer IV send their axons to layers II/III where they target both pyramidal neurons and interneurons (Thomson and Bannister 2003; Douglas and Martin 2004). Spiny stellate neurons are abundant in layer IV in the visual cortex of monkey and cat but in the visual cortex of rat, layer IV is dominated by pyramidal neurons (Peters and Yilmaz 1993). In cortical areas devoted to motor output this layer is thin. Layer IV is often divided into sub-layers, sometimes as many as four: IVA, IVB, IVCα, and IVCβ. Even though it is in this layer that the thalamic input terminates, 95% of the synapses in the visual cortex of cat are not from thalamic afferents but from corticocortical afferents (Martin 2002).

Layer V is mainly populated by pyramidal neurons, some of which are very large. These pyramidal neurons send their main bulk of axons to the white matter, but some axons are also sent to layers II/III where they target interneurons (Thomson and Bannister 2003; Douglas and Martin 2004). Efferent axons are also sent to the spinal cord and other deep structures. In rat, layer V pyramidal neurons receive excitatory input from all layers and inhibitory input from layers II/III (Schubert, Staiger et al. 2001). This layer is sometimes divided into two sub-layers: VA and VB.

Layer VI is on the edge to the white matter. The axons from the pyramidal neurons reach layers IV, V, and VI, and it is also in these layers where the main bulk of layer VI neurons' dendritic arborization is located (Zhang and Deschênes 1997). As in layer V there are efferent axons that go to the thalamic region (Rockel, Hiorns et al. 1980). About 50% of the neurons in this layer send afferents to other cortical areas (Zhang and Deschênes 1997). This layer is sometimes divided into two sub-layers: VIA and VIB.

To summarize (Salin and Bullier 1995; Braitenberg and Schuz 1998; Callaway 1998; Lamme, Supèr et al. 1998; Alonso 2002; Thomson and Bannister 2003; Douglas and Martin 2004), signals enter cortex via layer IV. The signals are then locally sent from layer IV to II/III and from layers II/III to V/VI. Pyramidal neurons in layers II/III project through the gray matter to other neurons in the same layer and we refer to these connections as intra cortical. Layers II/III pyramidal neurons also connect to layer IV neurons via the white matter and we refer to this projection as intercortical. Finally, the layers V/VI pyramidal neurons connect both inter- and intracortical to layers II/III and layer IV neurons and also within their own layers. These two layers also provide the output from cortex.

Layer I is not mentioned in the summary, but has been included into layers II/III. This means that there are three distinct layers in cortex; an upper layer with long-range recurrent and feed-forward projections, an input layer with short-range feed-forward and recurrent projections, and a deeper layer with output, recurrent, and feed-forward projections of long-range type. In an even more simplified functional view of cortex we only have two layers; a layer performing association consisting of layers II/III and layers V/VI, and a layer receiving input constituted of layer IV only.

Finally we note that there is a difference in the convergence/divergence of the forward and backward projections between lower and higher areas in the cortical processing hierarchy (Salin and Bullier 1995; Dancause, Barbay et al. 2005). In cat the forward projections from area 17 to area 18 innervate an area extending 1-3 mm (Price, Ferrer et al. 1994) and in the opposite direction the backward projections innervate a larger area extending 5-7 mm (Salin and Bullier 1995). It is generally conceived that the forward projections are convergent while the backward projections are divergent.

2.4.2 Neuron Distribution within Layers

The distribution of cortical depth between layers is shown in Table 2.1. It is worth noting that layer I, on average, constitutes 11% of the cortical depth but has very few neurons. Layers II/III and V/VI are anatomically similar. The anatomy of layer IV deviates slightly from that of layers II/III and V/VI.

The thickness of layers varies between different cortical areas and this can be explained by different needs for communication. In the motor and parietal areas that are devoted to output and association, layers III and V are slightly incremented and layer IV is reduced. In the visual cortex that processes visual inputs, layers III and IV have an increased thickness (Rockel, Hiorns et al. 1980).

The fractions of the total number of neurons for rat, cat, and macaque are tabulated in Table 2.2. Layers II/III are grouped into one, and no subdivisions of layers IV and VI are made. In a study of macaque the layers II/III, IV, and V-VI were examined, and in the somatosensory cortex the division is 40%, 20%, and 40% (Dombrowski, Hilgetag et al. 2001). In the motor cortex, the division is 40% in layers II/III and 60% in layers V-VI. In frontal area 10 of human and macaque, the neuron division between layers is identical (Semendeferi, Armstrong et al. 2001); 11% (I), 43% (II/IIII), 6% (IV), 40% (V/VI).

The average number of inhibitory (GABA-immunoreactive) neurons in the visual, somatosensory, and motor cortex of rat is in each area 15% (Beaulieu 1993). In macaque visual cortex, the figure is 20% (Beaulieu, Kisvarday et al. 1992). In Table 2.3 data are shown for a few different species, areas, and layers.

Table 2.1. Distribution of cortical depth between layers stated in percent. Legend: M1 – primary motor area; S1 – primary somatosensory area; V1 – primary visual area. References: (A) (Skoglund, Pascher et al. 1996); (B) (Beaulieu 1993); (C) (Beaulieu and Colonnier 1989); (D) (Peters and Yilmaz 1993); (E) (Beaulieu, Kisvarday et al. 1992); (F) (Peters and Sethares 1991).

	I	**II/III**	**IV**	**V**	**VI**
Rat, M1 (A)	11	22	10	28	29
Rat, M1 (B)	10	29	0	27	34
Rat, S1 (A)	10	23	12	24	31
Rat, S1 (B)	9	23	13	24	32
Rat, V2 (A)	16	24	13	22	25
Rat, V1 (B)	11	28	10	27	24
Cat, M1 (C)	15	37	0	16	32
Cat, V1 (D)	9	34	30	8	19
Average	11	28	11	22	28

Table 2.2. Distribution of neurons between layers stated in percent. Legend and references: see Table 2.1

	I	**II/III**	**IV**	**V**	**VI**
Rat, M1 (A)	1	28	14	24	33
Rat, M1 (B)	2	35	-	25	38
Rat, S1 (A)	1	26	23	17	33
Rat, S1 (B)	1	26	23	18	32

Rat, V2 (A)	1	32	18	19	30
Rat, V1 (B)	1	32	17	23	27
Cat, M1 (C)	3	51	-	14	32
Cat, V1 (D)	2	36	32	6	24
Macaque, V1 (E)	1	30	47	6	16
Macaque, V1 (F)	4	27	42	12	15
Average	2	32	22	16	28

Table 2.3. Percent inhibitory neurons in each layer. Legend and references: see Table 2.1

	I	**II/III**	**IV**	**V**	**VI**
Rat, M1 (B)	78	18	-	10	22
Rat, S1 (B)	87	15	11	18	25
Rat, V1 (B)	92	13	11	22	24
Cat, M1 (E)	85	21	19	25	18

2.5 Corticocortical Connections

There are two types of corticocortical connections; intercortical connections that go through the white matter and intracortical connections that spread horizontally within the cortex. The white matter is made up of four sorts of fibers (Braitenberg 2001): (i) afferents from subcortical structures; (ii) efferents to subcortical structures; (iii) commissural fibers that connects to the opposite hemisphere; (iv) association fibers connecting points in the same hemisphere.

Most fibers in the white matter are used for intra-hemispheric connections (Braitenberg and Schuz 1998). An estimated fraction of 10^{-2}-10^{-3} of the total number of fibers in the white matter are involved in subcortical projections (Braitenberg and Schuz 1998). The number of neurons that have commissural fibers has been estimated to be at most 2-3% and the number of commissural fibers is at least one order of magnitude smaller than the number of association fibers (Schüz and Preibl 1996).

2.5.1 Intracortical Connections

Intracortical connections that travel horizontally within the cortex are a general design principle of the cerebral cortex (Goldman and Nauta 1977). These lateral connections typically terminate in clusters of the size of hypercolumns (Goldman and Nauta 1977; DeFelipe, Conley et al. 1986; Gilbert and Wiesel 1989; Malach, Amir et al. 1993; Bosking, Zhang et al. 1997; Kisvarday, Toth et al. 1997; Yousef, Bonhoeffer et al. 1999; González-Burgos, Barrionuevo et al. 2000). In the visual cortex of rat, the number of excitatory synapses belonging to neurons in layers II/III that form local connections (within a radius of 300 μm) has been estimated to ~70% (Nicoll and Blakemore 1993). In the same study it was concluded that less than 1% of the synapses on neurons in layer V form local excitatory connections, meaning that most of the neurons in this layer are devoted to long-range communication. In other studies the short-range (<50 μm) connectivity between pyramidal neurons in layers II/III has been estimated to 0.1 (Holmgren, Harkany et al. 2003) and up to 0.8 (Hellwig 2000).

Lateral connections that stretch up to 7 mm in layers II/III (Galuske, Schlote et al. 2000) and somewhat shorter in layer V have been found in humans (Kenan-Vaknin, Ouaknine et al. 1992).

In a study of macaque visual cortex it is concluded that the lateral connections run up to a few millimeters (DeFelipe, Conley et al. 1986; Malach, Amir et al. 1993). The lateral connections are found both in layers III and V (DeFelipe, Conley et al. 1986). Lateral connections in layer III of macaque prefrontal cortex extend up to 2.2 mm (González-Burgos, Barrionuevo et al. 2000).

Axons from excitatory neurons in layers III and IV in cat visual cortex stretch out horizontally up to 3-4 mm (Kisvarday, Toth et al. 1997) and sometimes up to 6-8 mm (Gilbert and Wiesel 1989). Axons from inhibitory neurons stretch out horizontally up to 1.5-2 mm (Kisvarday, Toth et al. 1997). In lower layer IV of cat visual cortex, lateral connections extending up to 2-3 mm have been found (Yousef, Bonhoeffer et al. 1999).

In rat, lateral connections extending up to 2 mm have been found in both layers II/III and V (Telfeian and Connors 2003).

An interesting conclusion from the data presented is that the longest intra-cortical connections in humans and macaques do not seem to be longer than those in cat and rat. One explanation might be that the long-range intra-cortical connections in humans and macaques have not been found. Another possibility is that there might be physiological constraints preventing intra-cortical connections longer than a few millimeters. These constraints could be e.g. conduction delays and size constraints on the neuropil.

In summary, intra-cortical connections are made between sites located a few millimeters apart. Sites that are located further apart than a few millimeters, on the cortical surface, are connected by inter-cortical connections through the white matter.

2.5.2 Intercortical Connections and Cortical Areas

Cortex is divided into functionally specialized areas that are distinguished based on a variety of criteria and many different methods are used to analyze these criteria; morphology, pattern of connectivity, electro-physiological properties, and lesion studies (Kaas 1987). The differences in cytoarchitectonics between different cortical areas are explained by Rockel et al. (1980) as an effect of differences in the communication needs. The smallest cortical areas that have been found in mammals (shrews) are about 0.5 mm^2 (Catania, Lyon et al. 1999) (the size of one or a few hypercolumns). The connections between cortical areas are mediated by fibers going through the white matter. A first approximation of the cortical connectivity is that all areas are reciprocally connected (Shepherd and Koch 1998). In a review of area connectivity of cat and macaque it has been concluded that all major sensory systems and the motor system are organized hierarchically (Young, Scannell et al. 1994; Scannell, Blakemore et al. 1995). This does especially apply to the visual, auditory, somatosensory, and motor systems (Scannell, Burns et al. 1999).

In two literature studies, cat was determined to have 65 areas with 1139 connections and the macaque 72 areas with 758 connections (Young, Scannell et al. 1994; Scannell, Blakemore et al. 1995). No distinction between connections that have been actively looked for and reported absent, and connections that have not been studied was made in these two studies. The authors conclude that the number of connections found in macaque is very small and should probably be larger. In a more recent study (Scannell, Burns et al. 1999), 53 cortical areas in cat have been found and these are connected with 850 connections.

In a study of the macaque visual cortex (Jouve, Rosenstiehl et al. 1998), an interpolation technique was used to classify connections that have not been reported either existent or nonexistent. The interpolation was based on the shape of the known network of connections. Of a possible total of 900 connections, 647 connections are known between the 30 areas of the macaque visual cortex. Of the 647 known connections, 324 are known to exist and 323 are known not to exist. Of the unknown connections, the interpolation algorithm classified 32 as nonexistent and the remaining 191 connections as existent. In summary, many of the unknown connections were assumed to exist. This result highlights an issue of controversy in the analysis by Young et al. (1994) and Scanell et al. (1995), where unknown connections are treated as absent. Scanell et al. (1999) provide five arguments for treating unknown connections as absent, but the main argument is that the unknown connections, in their view, have a very small effect on cortical function because they are weak per definition.

It has been hypothesized that the human cortex has about 110 areas (Kaas 1987) and the mouse cortex has been estimated to have 24 areas (Krubitzer 1995).

2.6 Visual Cortex

Much work has been done trying to characterize the primary visual cortex, V1 (Callaway 1998). There are several good reasons to why V1 has been intensively studied; vision is an important modality for many mammals, V1 occupies a large cortical area, and it is the first stage for processing of visual information in cortex. Because it is the first stage in cortical visual processing, there is a strong correlation between neuronal activity and a presented visual stimulus. This makes it possible to assume a causal relation between visual stimuli and neural activity, which in turn means that the workings of these neurons can be experimentally investigated.

In the late 1950's Hubel and Wiesel were pioneers in studying the functional properties of V1. Up until then, much work had been done trying to characterize the retina and lateral geniculate nucleus (LGN), a ganglia located in thalamus that the visual stimuli passes through on its way to V1. Hubel and Wiesel found that neurons in V1 react selectively to single spots of light, i.e. their response properties can either be *on center – off surround* or *off center – on surround*. These neurons were called *simple cells*. Further, they found neurons that respond to line segments, called *complex cells*, and line endings, called *hypercomplex cells*, in V1. Their experiments also showed that the response properties of neurons were similar when cortex was traversed vertically with an electrode (Hubel and Wiesel 1963). This formed the basis for the idea of a columnar organization of cortex perpendicular to its surface. The orientation selectivity of neurons in V1 was a surprising discovery because no morphological methods, that were then available, had given any hint about the orientation columns of V1 (Hubel and Wiesel 1977). Further, Hubel and Wiesel found that by traversing about 0.5-1 mm of cortex, one or several full rotations through 180° are accomplished, which is also supported by more recent experiments (Blasdel 1992b). The change in orientation preference, in Hubel and Wiesel experiments, occurred in small steps approximately every 50 μm. A step corresponded to approximately a 10° change in orientation preference. Recent experiments with microelectrode-arrays and staining techniques, where large areas of the cortical surface are mapped, have shown that the pattern of iso-orientation lines form an irregular map of pinwheels (Bartfeld and Grinvald 1992; Blasdel 1992a; Hübner, Shoham et al. 1997). The iso-orientation lines come together in what is called singularities, which form the center points of pinwheels. Interestingly, the crossings between the iso-orientation lines and the borders of ocular dominance columns occur in approximately right angels.

A weak effect of orientation tuning is present in layer IV, but it is much stronger in layers II/III (Cürüklü 2005). It is generally conceived that this orientation tuning is implemented by the projection from layer IV to II/III and also supported by lateral projections in these receiving layers (Alonso and Martinez 1998; Somers, Todorov et al. 1998).

Hubel and Wiesel found that there are at least two columnar systems in V1: ocular dominance columns and orientation columns. Today it is also known that there are columnar systems with neurons that are sensitive to spatial frequencies (Hübner, Shoham et al. 1997; Issa, Trepel et al. 2000) and color in the visual input (Dow 2002). A common trait of all of these columnar systems is that they are approximately of the same size, i.e. 0.5-1 mm, and they are all seen to perform some sort of normalization of the neurons' activities within the columns (Hubel and Wiesel 1977; Hübner, Shoham et al. 1997). For a review of columnar systems see Dow (2002).

Columns in the visual cortex can be visualized by a staining technique called the cytochrome oxidase (2-deoxyglucose) method. When used, this method reveals dark spots where the metabolic activity is slightly increased. These dark spots are called *blobs* and between these spots there are larger and lighter colored regions that are called *interblobs*. The blobs have a diameter of ~150 μm and a repeat period of 350-550 μm and they are thought to be involved in the coding of color information (Mountcastle 1997; Dow 2002). Neurons in the blob regions receive more thalamic afferents than those in the inter-blob regions (Mountcastle 1997). It has been suggested that the blobs are located in the singularities formed by the orientation selective neurons (Dow 2002). Blobs have been found in many types of mammals, but primarily in primates (Murphy, Jones et al. 1995).

Hubel and Wiesel preformed their experiments on both cats and macaque monkeys. The results of their experiments were very similar for both of these two species. In their paper from 1977 they discuss if other mammals also have a columnar organization in the visual cortex. In smaller mammals such as rat, it does exist orientation sensitive neurons, but these are not ordered in regular patterns as in macaques and cats (Girman, Sauvé et al. 1999). There is no smooth transition of preferred orientation between nearby neurons in rat. In the prosimian, Galago, both orientation and ocular dominance columns have been found (Hubel and Wiesel 1977).

Hubel and Wiesel proposed the idea that the visual cortex was built up out of small, repetitive, building blocks. They hypothesized that these building blocks contained both an ocular dominance and an orientation column. This idea is formally expressed in their ice cube model, where ocular dominance and orientation columns together form 1-2 mm squares of cortical modules. This hypothesis has been criticized by some of being too simplistic (Swindale 1998).

2.7 Cortical Columns

The cortex of mammals is not only organized horizontally, in layers, but also vertically in columnar sections. Evidence of the columnar structure is available from many different sources: During the development of cortex the cell mitosis occurs in columns. In the mature cortex neurons are seen to cluster in vertical columns. Both afferent connections to cortex and corticocortical connections terminate in clusters, i.e. they have a patchy layout.

Electrophysiological and metabolic studies show that neurons are activated in small patches that are spread out over the cortical surface.

What the functional properties are of cortical columns is a question that is intensely debated. Some believe that the columnar structure of cortex simply is an artifact from development. We approach the problem by looking at what is known about the cortical columns in the visual cortex, and hence we use the term *hypercolumn* to denote the larger cortical columns and *minicolumn* to denote the smaller cortical columns.

Our view of cortex is similar to that of Szentágothai and Eccles, who have suggested that the hypercolumns are modules in a cortical associative network (Szentágothai 1975; Eccles 1981). More recently, the minicolumn has been proposed as the functional unit of cortex based on arguments of redundancy, connectivity, and processing time (Lücke and Malsburg 2004).

Based on their size, columns are divided into two categories; large hypercolumns and small minicolumns. Hypercolumns can both be rectangular and circular. Typically they have a size of 400-600 μm, but this figure can vary between 200-2000 μm. The minicolumns are typically considered to be small circular columns with a diameter between 20-60 μm.

In the literature, the hypercolumn is referred to as cortical microcircuits, cortical column, cortical module, macrocolumn, barrel, or segregate. The term "hypercolumn" originates from Hubel and Wiesel and their experiments in the visual cortex and they used the term to denote both the ocular dominance columns and the orientation columns. They motivated the use of a common name by the fact that these two structures have a similar size. Segregates comes from Favorov and Diamond (1990) and their experiments in the somatosensory cortex. In the somatosensory area in e.g. mouse, the hypercolumns are called barrels (Woolsey and Loos 1970). Minicolumns are sometimes referred to as microcolumns.

Two excellent reviews on the topic cortical columns are provided by Mountcastle (1997) and Buxhoeveden and Casanova (2002).

2.7.1 Developmental and Evolutionary Aspects

The neurons that form the cortex are not created within the cortex; instead they are generated in the cerebral ventricle. In the cerebral ventricle there are progenitor cells and glia cells ordered in small zones called radial units. From each radial unit, the glia cells stretch out to the forming cortical plate. The neurons are generated from the progenitor cells and then migrate radially outwards to the cortical plate along the stretched out glia cells. Upon reaching the cortical plate the neurons form small columns, called ontogenetic columns. Approximately 90% of the neurons stay in the ontogenetic column corresponding to the radial unit in which they were born (Mountcastle 1997). The developmental period, in which a neuron is born, determines its final laminar position in the mature cortex (Rakic 1974; Rakic 1995).

Rakic (1995) states that the number of ontogenetic columns determines the area of the cortical surface, whereas the number of cells within the columns determines the thickness of cortex. This would allow for a simple way of expanding the cortex, simply by producing more progenitor cells and hence more columns.

Both Mountcastle (1997) and Buxhoeveden and Casanova (2002) argue that ontogenetic columns are what become minicolumns in the adult cortex. This is motivated by anatomical studies of development in human auditory cortex, which conclude that ontogenetic radial cell columns are the same vertical minicolumns found in the adult brain (Buxhoeveden and

Casanova 2002). Similar results have also been found in the somatosensory area of the developing rat cortex.

The hypothesis that the ontogenetic columns are what form the minicolumns is in the same line of thinking as that of Hubel and Wiesel. They argued that the hypercolumns to a large extent are genetically programmed (Hubel and Wiesel 1977). Lesion studies have validated the idea that the columns are genetically preprogrammed. But it is vital for the developing columnar structures to receive input and be allowed to have normal activity. If the activity is manipulated in the developing columns, the adult columns do not function normally (Sur, Angelucci et al. 1999).

Mountcastle points out that it is remarkable that the size of columns has been preserved in the course of evolution. Primitive primates like the tree shrew, with a small prefrontal cortex, have columns that are of the same size as that of macaque monkeys, which have a ten times larger prefrontal cortical surface area. He goes on to state that this is compatible with the idea that cortex has expanded during evolution by adding new columns of a constant size (Mountcastle 1997).

There are large differences in the cortical surface area between different mammals (Hofman 1985). The ratio between the cortex area in mouse, macaque and human is approximately 1:100:1000, but yet the thickness only vary a factor 4. This means that cortex is enlarged through an increase of its area and not its thickness (Rakic 1995). Furthermore, it means that the number of minicolumns, and not their size, is increased when cortex expands. This has been seen in transgenic mice (Chenn and Walsh 2002).

In humans the ontogenetic columns are very small during their initial development; 4.6 µm at 14 weeks, 9.6 at 29 weeks, and 13 µm at 42 weeks (Buxhoeveden and Casanova 2002). In humans, a fully developed ontogenetic column is approximated to have 100 neurons (Rakic 1995).

2.7.2 Minicolumns

In Mountcastle's experiments (1957) he found that the neural activity evoked in response to a somatosensory stimuli occurred in all layers and not only in layer IV. He concluded that this was an artifact from the minicolumns discovered in the somatosensory cortex by Lorente de Nó in 1938 (Mountcastle 1957; Buxhoeveden and Casanova 2002). Lorente de Nó studied the cylindrical columns of increased cell densities resulting from the ontogenetic columns (section 2.7.1). Today the minicolumn has been found in the cortex of most mammals both as an anatomical and functional structure (Mountcastle 1997; Buxhoeveden and Casanova 2002).

The horizontal diameter of a minicolumn varies slightly between different cortical areas and mammalian species. The typical minicolumn has a total diameter of about 50 µm and an inner circle with a diameter of approximately 30 µm where the neuron density is high (Buxhoeveden, Switala et al. 2000). In mammals minicolumns typically have a diameter of about 20-60 µm (Changeux 1985; Leise 1990; Calvin 1995; Mountcastle 1997; Braitenberg and Schuz 1998; Buxhoeveden, Switala et al. 2000; Buxhoeveden and Casanova 2002), in humans the diameter is about 40-50 µm (Buldyrev, Cruz et al. 2000; Buxhoeveden, Switala et al. 2001; Buxhoeveden and Casanova 2002), and about 36 µm in macaque (Buxhoeveden, Switala et al. 2001).

The number of neurons in a minicolumn has been estimated to; 80-100 (Mountcastle 1997; Buxhoeveden and Casanova 2002), 100 (Calvin 1995; Rakic 1995), 110 (Changeux 1985), and 80-260 (Leise 1990). There are studies indicating that the neuron count in minicolumns located in primates' V1 is above 200 (Rockel, Hiorns et al. 1980; Mountcastle 1997), but studies in macaques have given a mean value of 142 neurons per minicolumn (Peters and Sethares 1991). In a study of rat, the average number of neurons in a cortical minicolumn has been measured to 72 in the visual cortex, 90 in the somatosensory cortex, and 71 in the motor cortex (Beaulieu 1993).

In Hubel and Wiesel experiments, electrophysiological evidence of minicolumns was seen as small stepwise changes in orientation preference every 20-50 μm (Hubel and Wiesel 1977). The spatial precision in the positioning of their electrodes was ±20 μm.

Minicolumns can also be defined by the apical dendritic clusters of layer V pyramidal neurons that ascend through layer IV into layers II/III. These clusters have a diameter of about 30 μm with a center-to-center spacing of 20-80 μm. These dendritic clusters are very stereotypical across different cortical areas and species (Feldman and Peters 1974; Silberberg, Gupta et al. 2002) and they have been found in the visual cortex of both rats (55-60μm), cats, and monkeys (30 μm) (Feldman and Peters 1974; Peters and Kara 1987; Peters and Sethares 1991; Peters and Yilmaz 1993; Peters and Sethares 1996). Mountcastle (1997) refers to 15 studies where the minicolumn diameter has been measured based on the size of the apical dendrites of layer V neurons, and conclude that the overall mean diameter is 56 μm. In these studies data were collected from mice, rats, rabbits, cats, and monkeys, with a variety of methods.

The average number of apical dendritic clusters (minicolumns) per mm^2 in the visual cortex of macaque is 1270 (Peters and Sethares 1991), only 359 in the visual cortex of rat (Peters and Kara 1987), and 401 in the visual cortex of cat (Peters and Yilmaz 1993). These are the expected counts if the minicolumns are tightly packed and have a diameter of ~30 μm, ~60 μm, and ~55 μm respectively.

In layer II and upper layer III there are double bouquet neurons that send tightly packed bundles of descending axons down to layer V (Jones 2000). These bundles are ~10 μm in diameter and have a periodicity of 15-30 μm (DeFelipe, Hendry et al. 1990). The columnar arrays formed by these double bouquet neurons are clearly visible in primates but less apparent in rodents.

Each minicolumn has afferent input, efferent output, and intra-cortical circuitry (Buxhoeveden, Switala et al. 2000). Within each minicolumn there are both excitatory pyramidal neurons and inhibitory interneurons (Buxhoeveden, Switala et al. 2000). Although there exists some differences in the exact layout and wiring between minicolumns located in different parts of the cortex (such as the exact size and structure and active neurotransmitter) it seems as though it represents a general building block of the cortex (Silberberg, Gupta et al. 2002; Mountcastle 2003).

A nerve regeneration experiment done with monkeys has provided evidence of the generic nature of minicolumns. In this experiment, the receptive fields of a number of minicolumns in the somatosensory cortex were mapped. These minicolumns had their receptive fields associated with regions on the monkey's hand. The nerves leading from the hand was then cut. Recordings taken from the same minicolumns at a later point in time when the nerves to the hand had regenerated showed that the receptive fields had been remapped (Mountcastle 1997).

The lower bound of the minicolumn diameter is 10 μm, because this is approximately the space occupied by a single neuron (Buldyrev, Cruz et al. 2000). A study in rat has shown that the average size of a neuron's soma is between 10-15 μm, and the smaller values are attained for inhibitory neurons (Beaulieu 1993).

An interesting observation is that the human minicolumn contains two times more neuropil space than those in macaque (Buxhoeveden, Switala et al. 2001). This could reflect a need for more wiring, i.e. corticocortical connections, in the human brain to maintain the same level of connectivity as in the smaller brain of the macaque.

2.7.3 Hypercolumns

A hypercolumn is a generic structure seen in the cortex of mammals (Hubel and Wiesel 1977; Changeux 1985; Leise 1990; Calvin 1995; Mountcastle 1997; Shepherd and Koch 1998; Buxhoeveden and Casanova 2002). Similar to minicolumns, hypercolumns stretch vertically through the layers of cortex and they can either have a rectangular or circular shape. The typical diameter of a hypercolumn is 300-600 µm (Mountcastle 1997), but sometimes the length of rectangular hypercolumns can extend up to 3000 µm (Leise 1990).

In Mountcastle's pioneering experiment in 1957, electrodes were inserted into the somatosensory cortex of cat. When an electrode was inserted perpendicular to the cortical surface, neurons in a small columnar volume (minicolumn) showed homogenous response properties. These neurons were only activated when a stimulus was present in their particular receptive field. The experiment also indicated that these minicolumns are organized into hypercolumns and exhibit mutual inhibition. The experiment did not reveal the exact size of the hypercolumns, but confined it to less than 500 µm.

Hypercolumns are identified based on their electrophysiological and metabolic properties, and the anatomical layout of horizontal connections in layers II/III and thalamic afferents (Mountcastle 1997). Opposed to minicolumns, hypercolumns are typically not defined by shifting cell densities with an exception made for the barrel cortex of mouse and rat (Woolsey and Loos 1970; Feldman and Peters 1974). In studies of ferrets, the hypercolumnar structure is developed during the first postnatal weeks and it is hypothesized to be a product of neural activity (Ruthazer and Stryker 1996).

In the visual cortex the rectangular ocular dominance columns are approximately 400 µm wide and a few 1000 µm long, and the oval or hexagonal orientation columns are 400-1000 µm wide (Hubel and Wiesel 1977; Bartfeld and Grinvald 1992; Mountcastle 1997; Amirikian and Georgopoulos 2003; Lund, Angelucci et al. 2003).

Favorov and Diamond (1990) did experiments with grids of closely spaced microelectrodes in the somatosensory cortex of cat. Their results showed that a hypercolumn is a hexagonal structure, and it is typically surrounded by six other hypercolumns. The width of the columns found in their study was 300-400 µm.

In the somatosensory area of mouse, there is anatomical evidence in layer IV of columns (Woolsey and Loos 1970). These columns are 100-400 µm wide. Studies of the layer IV to III feed-forward projections in rat somatosensory cortex show that these connections are confined within an area of less than 400 µm in diameter (Bender, Rangel et al. 2003).

Electrophysiological studies have revealed that there exists a columnar structure similar to that of ocular dominance columns also in the auditory cortex. Penetrations made parallel to the cortical surface, in auditory cortex, have shown patchy blocks of neurons that respond to different sound frequencies. These structures have been found in dogs, cats, monkeys, and also in other mammals (Mountcastle 1997).

In associative cortical areas, the input comes from a wide range of sources. It is therefore not possible to control the input to these areas, which makes it difficult to experimentally investigate these areas. The strongest evidence of a hypercolumnar structure in the associative cortical areas comes from the fact that the axonal arborization occurs in patches 300-500 μm wide.

Lesie (1990) suggests that the hypercolumns are an evolutionary continuation of the nuclei and ganglia structures, which are present in both vertebrate and invertebrates.

2.7.4 The Hypercolumn as a Module of Minicolumns

A conclusion from both metabolic and electrophysiological studies, on both cats and monkeys, is that a hypercolumn contains a number of minicolumns (Mountcastle 1997; Amirikian and Georgopoulos 2003). The exact belonging of a minicolumn may be somewhat diffuse and it is conceivable that a minicolumn could take part in several hypercolumns. The typical number of minicolumns in a hypercolumn has been estimated to 60-80 (Buxhoeveden and Casanova 2002) and 80 (Mountcastle 1997).

Much of the experimental data indicate that the minicolumns within a hypercolumn have a functional dependency (Mountcastle 1997). Buxhoeveden and Casanova (2002) have stated that the output of a hypercolumn results from tightly knit interactions between the smaller minicolumns within it. Hubel and Wiesel showed by electrophysiological experiments in primates that the hypercolumn function as a WTA (winner-take-all) circuitry in the visual cortex (Hubel and Wiesel 1977). Similar types of interactions have also been found within a hypercolumn in the motor cortex of monkeys (Amirikian and Georgopoulos 2003).

It has been suggested that large basket neurons are responsible for the orientation tuning (normalization) in the hypercolumns (Szentágothai 1975; Cürüklü and Lansner 2002).

2.8 The Size of Cortex

The numbers relating to cortical micro- and macro-architecture vary a lot. This variation is due to uncertainties in the measurements, but also due to large variations between individuals e.g. the size of V1 can vary with a factor 2 (Andrews, Halpern et al. 1997; Pakkenberg and Gundersen 1997). Neuron counts for the entire cortex are generally extrapolations of pinpoint counts and hence these counts are affected by both the particular area from which the sample was taken and tissue shrinkage. The volume shrinkage can sometimes be as high as 80% (Buxhoeveden, Switala et al. 2000).

2.8.1 Volume of Gray and White Matter

In this section we have gather data on the volume of gray and white matter. In summary, the ratio of gray / white matter is 1 in human, 2.5 in macaque, 3 in cat, 6.5 in rat and mouse. This means that there is relatively less white matter in smaller mammals.

- In human, the volume of gray matter in cortex has been measured to 489 cm^2 and the volume of gray in nuclei to 42 cm^2 (Pakkenberg and Gundersen 1997). The total volume of gray in the cerebrum has been measured to 683 cm^2 (Hofman 1988). The volume of white matter in the cerebrum has been measured to 451 cm^2 (Pakkenberg and Gundersen 1997) and 484 cm^2 (Hofman 1988).

- In macaque, the volume of gray matter has been measured to 55 cm^2 and the volume of white matter to 21 cm^2 (Hofman 1988).

- In cat, the volume of gray matter has been measured to 15 cm^2 (Hofman 1988) and 14 cm^2 (Mayhew, Mwamengele et al. 1990). The volume of white matter has been measured to 5 cm^2 (Hofman 1988).
- In rat, the volumes of gray and white matter have been estimated to 0.4 cm^2 and 0.06 cm^2 (Zhang and Sejnowski 2000).
- In mouse, the volumes of gray and white matter have been estimated to 0.1 cm^2 and 0.015 cm^2 (Zhang and Sejnowski 2000).

2.8.2 Cortical Area

A compilation of data on the cortical surface area is presented in the following bullets.

- The total cortical surface in human has been estimated to 2200 cm^2 (Changeux 1985) and 2400 cm^2 (Hofman 1985; Nieuwenhuys, Donkelaar et al. 1997). In a study of 94 subjects, the neocortex was measured to 1820 cm^2 (Pakkenberg and Gundersen 1997).
- The total cortical surface in macaque has been estimated to 250 cm^2 (Hofman 1985).
- The total cortical surface in cat has been estimated to 83 cm^2 (Hofman 1985; Nieuwenhuys, Donkelaar et al. 1997) and measured to 89.8 cm^2 (Mayhew, Mwamengele et al. 1990).
- The total cortical surface in rat has been estimated to 4-5 cm^2 (Changeux 1985), 6 cm^2 (Nieuwenhuys, Donkelaar et al. 1997), and 6.5 cm^2 (Hofman 1985).
- In mouse, the total cortical surface area is 2.2 cm^2 and that of neocortex is 1.4 cm^2. These values were computed based on data from (Braitenberg and Schuz 1998).

2.8.3 Cortical Thickness

In the following bullets we list data on the cortical thickness. In Table 2.4 we have compiled data on the cortical thickness from Rockel et al. (1980).

- The overall thickness of the human cortex has been measured to 2.69 mm (Pakkenberg and Gundersen 1997) and estimated to 2.8 mm (Hofman 1988; Nieuwenhuys, Donkelaar et al. 1997). Measurements taken with magnetic resonance imaging show an average cortical thickness of 2.5 mm and a thickness of ~3 mm in association areas (Fischl and Dale 2000). A cortical thickness of 2.5 mm has been measured in the middle frontal cortex (Scheff, DeKosky et al. 1990; Scheff, Price et al. 2001).
- In the visual cortex of macaque, the cortical depth has been measured to 1.6 mm (Peters and Sethares 1991). The average thickness of the macaque cortex has been estimated to 2.2 mm (Hofman 1988).
- The overall thickness of the cat cortex has been measured to 1.34 mm (Peters and Yilmaz 1993), 1.8 mm (Hofman 1988; Nieuwenhuys, Donkelaar et al. 1997), and in the motor cortex to 1.7 mm (Beaulieu and Colonnier 1989).
- In rat, the cortical thickness has been measured to 1.9 mm in the motor cortex, 1.9 mm in the somatosensory cortex, and 1.4 mm in the visual cortex (Beaulieu 1993; Skoglund, Pascher et al. 1996). In the frontal cortex, the thickness has been measured to 2.1 mm

(Beaulieu 1993). The cortical thickness in somatosensory cortex has been measured to 1.3 mm (Brunso-Bechtold, Linville et al. 2000).

- The average cortical thickness in mouse has been estimated to 0.8-0.9 mm (Braitenberg and Schuz 1998).

Table 2.4. The data on cortical thickness is a compilation from Rockel et al. (1980). These measurements were not corrected for shrinkage.

	Motor area 4	Parietal area 7	Visual area 17	Mean thickness (mm)
Mouse	1,0	0,6	0,7	0,8
Rat	1,7	1,0	1,0	1,2
Cat	1,7	1,4	1,4	1,5
Macaque	2,5	1,8	1,5	1,9
Human	3,1	2,9	1,9	2,6

2.8.4 Neuron Numbers

Neuron counts have been done in the motor, somatosensory, frontal, temporal, parietal, and visual areas in mouse, rat, cat, macaque, and humans (Rockel, Hiorns et al. 1980). The results show that the neuron density per area (30×30 µm) is roughly constant (100-110 neurons) for all areas and species except for the visual cortex (260-270 neurons) of primates where the density per area is 2.5 times higher. These counts were centered on and performed in an area corresponding to the size of a minicolumn. The authors conclude that if the counts had been done over a wider area, the figures corresponding to a 30×30 µm area section would have been 80-90 neurons instead of 100-110. The counts were not corrected for shrinkage, and a count of 80-90 neurons corresponds to ~$9.5 \cdot 10^4$ neurons·mm^{-2} and 100-110 neurons corresponds to ~$1.2 \cdot 10^5$ neurons·mm^{-2}. If the neuron counts in the visual areas are reduced by 20% (to correct for the small area), the area density is ~$2.4 \cdot 10^5$ neurons·mm^{-2}. The results by Rockel et al. (1980) have been criticized by many for the methodological errors but their conclusion, that the number of neurons beneath a unit area is roughly constant, is still valid (Harrison, Hof et al. 2002).

In Figure 2.2 we have compiled the data of the neuron densities. This compilation shows the large variation in the data.

- In a thorough study, the mean number of neurons in the human cortex of 94 subjects was measured to $2.28 \cdot 10^{10}$ in males, $1.93 \cdot 10^{10}$ in females, and an average of $2.15 \cdot 10^{10}$ (Pakkenberg and Gundersen 1997). The neuron density has been measured to $2.8 \cdot 10^4$ neurons·mm^{-3} (Haug 1987) and $4.4 \cdot 10^4$ neurons·mm^{-3} (Pakkenberg and Gundersen 1997). In area 10 of humans, the neuron density has been measured to $3.4 \cdot 10^4$ neurons·mm^{-3}, and in apes it has been measured to $4.7\text{-}8.6 \cdot 10^4$ neurons·mm^{-3} (Semendeferi, Armstrong et al. 2001).

- The total number of neurons in the macaque neocortex has been measured to $1.0 \cdot 10^9$ and the neuron density was ranging between $7.2 \cdot 10^4\text{-}1.4 \cdot 10^5$ neurons·mm^{-3} (Mouton, Price et al. 1997). The average neuron density in macaque cortex has been estimated to $3.8 \cdot 10^4$ neurons·mm^{-3} (Haug 1987). The average neuron density in the macaque visual cortex has been measured to $1.148 \cdot 10^5 \pm 5200$ neurons·mm^{-3} and the area density to $1.744 \cdot 10^5 \pm 3000$ neurons·mm^{-2} (Beaulieu, Kisvarday et al. 1992). The neuron density in the different areas of the macaque prefrontal cortex is ranging from $3.9\text{-}5.9\cdot10^4$ neurons·mm^{-3} (Dombrowski, Hilgetag et al. 2001). The area density in the prefrontal cortex has been measured to $\sim9.8\cdot10^5$ neurons·mm^{-2} (Dombrowski, Hilgetag et al. 2001).

- In cat visual cortex, the neuron density has been measured to $5\cdot10^4$ neurons·mm^{-3} (Beaulieu, Kisvarday et al. 1992). The average neuron density in cat cortex has been estimated to $5.0\cdot10^4$ neurons·mm^{-3} (Haug 1987). The average area density of neurons has been measured to $8.15\cdot10^4$ neurons·mm^{-2} (Peters and Yilmaz 1993). In the motor cortex of cat, the neuron density has been measured to $\sim2.9\cdot10^4$ neurons·mm^{-3} and the density per area to $\sim4.9\cdot10^4$ neurons·mm^{-2} (Beaulieu and Colonnier 1989).
- In rat visual cortex, the neuron density has been estimated to $8\cdot10^4$ neurons·mm^{-3} (Beaulieu, Kisvarday et al. 1992), $5.2\cdot10^4$ neurons·mm^{-3} (Beaulieu 1993), $7.4\text{-}8.8\cdot10^4$ neurons·mm^{-3} (Turner and Greenough 1985), and $7.7\cdot10^4$ neurons·mm^{-3} (Skoglund, Pascher et al. 1996). In the motor cortex, the neuron density has been measured to $4.9\cdot10^4$ neurons·mm^{-3} (Skoglund, Pascher et al. 1996) and $3.4\cdot10^4$ neurons·mm^{-3} (Beaulieu 1993). In the somatosensory cortex, the neuron density has been measured to $6.9\cdot10^4$ neurons·mm^{-3} (Skoglund, Pascher et al. 1996) and $4.8\cdot10^4$ neurons·mm^{-3} (Beaulieu 1993). The area densities in rat motor, somatosensory, and visual cortex have been measured to $9.1\cdot10^4$, $1.34\cdot10^5$, and $1.06\cdot10^5$ neurons·mm^{-2} (Skoglund, Pascher et al. 1996). The neuron density in layers II-IV in the visual cortex of rat has been measured to $8.7\cdot10^4$ neurons·mm^{-3} and for all layers $7.2\cdot10^4$ neurons·mm^{-3} (Miki, Fukui et al. 1997). The average neuron density in rat neocortex has been measured to $8.8\text{-}9.6\cdot10^4$ neurons·mm^{-3} and the total number of neurons to $1.5\text{-}1.6\cdot10^7$ (Mooney, Napper et al. 1996). The average neuron density in rat cortex has been estimated to $4.4\cdot10^4$ neurons·mm^{-3} (Haug 1987). The total number neurons in the rat cortex has been measured to $2.1\cdot10^7$ (Korbo, Pakkenberg et al. 1990). The neuron densities in 40 rats were measured and the values ranged between $6.5\cdot10^4\text{-}1.15\cdot10^5$ neurons·mm^{-3}. The average neuron density was $8\cdot10^4$ neurons·mm^{-3} in all areas except the visual cortex where it was $1.06\cdot10^5$ neurons·mm^{-3} (Korbo, Pakkenberg et al. 1990).
- The total number of neurons in mouse cortex has been estimated to $1.6\cdot10^7$ and in the neocortex to 10^7 (Braitenberg and Schuz 1998). The average density has been estimated to $9.2\cdot10^4$ neurons·mm^{-3} (Braitenberg and Schuz 1998) and $8.2\cdot10^4$ neurons·mm^{-3} (Haug 1987). In the entire brain of mouse there are $\sim7.5\cdot10^7$ neurons and $3.5\cdot10^7$ of these are located in the cerebellum (Williams 2000). We estimate that there are about $1.5\cdot10^7$ neurons in the brainstem, thalamus and other none cortex structures. This leads to a total number of neurons in the mouse cortex of $\sim2.5\cdot10^7$.

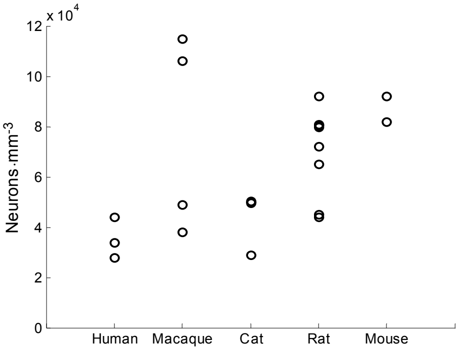


Figure 2.2. The mean neuron density in each study (circles) plotted for each species. Some of the outliers are due to measurements restricted to the visual cortex (macaque). A trend towards lower neuron densities in larger mammals is present.

2.8.5 Synaptic Numbers

In this section we list the data we have gather on the number of synapses in cortex. Instead of directly measuring the number of synapses, the number of synaptic terminals, boutons, is sometimes measured. The relationship between the number of synapses and that of synaptic boutons is somewhat uncertain. Mouton et al. (1997) state that there are 1-2 synapses for each bouton.

In Figure 2.3, the data on synaptic densities have been compiled for a number of different mammalian species. The variation in this data is large, as for the neuron density data.

- The average number of synapses in human is $\sim 1.5 \cdot 10^{14}$ (Pakkenberg, Pelvig et al. 2003) and the number of synapses per neuron has been measured to 7500. The synaptic density in human cortex has been measured to $1.5\text{-}1.6 \cdot 10^{9}$ synapses·mm^{-3} (DeFelipe, Marco et al. 1999). The synaptic density in layer III has been measured to $3.8 \cdot 10^{8}$ synapses·mm^{-3} (Scheff, DeKosky et al. 1990) and $5.2 \cdot 10^{8}$ synapses·mm^{-3} (Scheff, Price et al. 2001). The synaptic density in layer V has been measured to $2.8 \cdot 10^{8}$ synapses·mm^{-3} (Scheff, DeKosky et al. 1990) and $3.8 \cdot 10^{8}$ synapses·mm^{-3} (Scheff, Price et al. 2001). In an old study from 1979 it is stated that the mean synaptic density is $1.1 \cdot 10^{9}$ synapses·mm^{-3} and the number of synapses per neuron is $8.4 \cdot 10^{4}$, which is extremely high (Huttenlocher 1979).

- The average number of synapses per neuron in the visual cortex of macaque has been measured to $4.5 \cdot 10^{8}$ (Beaulieu, Kisvarday et al. 1992), $4\text{-}5 \cdot 10^{8}$ (Bourgeois and Rakic 1993) and the average number of synapses per neuron to ~3900 (Beaulieu, Kisvarday et al. 1992). The total number of synaptic boutons in the neocortex of macaque has been measured to $1.3 \cdot 10^{12}$ and the density of these synaptic boutons is ranging from $1.2 \cdot 10^{8}$-$2.2 \cdot 10^{8}$ boutons·mm^{-3} (Mouton, Price et al. 1997).

- In cat visual cortex the synapse density has been measured to $2.9 \cdot 10^{8}$ synapses·mm^{-3} and the average number of synapses per neuron to ~5700 (Beaulieu, Kisvarday et al. 1992). In the motor cortex of cat, the synaptic density has been measured to $2.6\text{-}2.9 \cdot 10^{8}$ synapses·mm^{-3} and the average number of synapses per neuron to ~8000 (Beaulieu and Colonnier 1989).

- In rat visual cortex the synapse density has been estimated to $7.5\text{-}9.5\cdot10^8$ synapses·mm^{-3} and the average number of synapses per neuron to ~10^4 (Beaulieu, Kisvarday et al. 1992). The synaptic density in layers II-IV in the visual cortex of rat has been measured to $9.2\cdot10^8$ synapses·mm^{-3}, and for all layers $7.3\cdot10^8$ synapses·mm^{-3} (Miki, Fukui et al. 1997). The synaptic density in layers I-IV in the visual cortex of rat has been measured to $5.6\text{-}6.9\cdot10^8$ synapses·mm^{-3} and the number of synapses per neuron to 6500-9500 (Turner and Greenough 1985). The synaptic density in the parietal cortex of rat has been measured to $2.2\cdot10^9$ synapses·mm^{-3} (Calverley, Bedi et al. 1988). The average synaptic density in rat cortex has been measured to $4.4\text{-}5.9\cdot10^8$ synapses·mm^{-3} (Wong, Campbell et al. 1998). The synaptic density in somatosensory cortex was measured to $9\cdot10^8$ synapses·mm^{-3} in layer II and 7.5 synapses·mm^{-3} in layer IV (Brunso-Bechtold, Linville et al. 2000). The number of synapses per neuron was 9500 in layer II and 6500 in layer IV (Brunso-Bechtold, Linville et al. 2000).
- In mouse cortex the average density of synapses has been measured to $7.2\cdot10^8$ synapses·mm^{-3} and the total number of synapses to 10^{11} (Braitenberg and Schuz 1998). The number of synapses per neuron has been estimated to 7800.

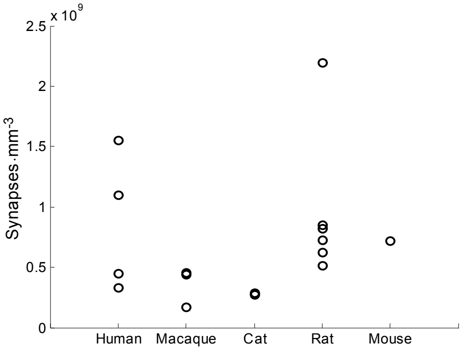


Figure 2.3. The mean synaptic density in each study (circles) plotted for each species.

2.8.6 Summary of Anatomical Data

There are considerable variations in the data, even in measurements taken from one particular area, in one particular species. Although, our data support the hypothesis of a constant neuron density per area between areas and species (Table 2.6). The data do not give any hint in the direction of a constant synaptic density between species. Concerning the issue of number of synapses per neuron, no exact number can be read out from the data, and the case of a constant number of 8000 synapses per neuron across all mammal species lies within the range of what can be supported by the data. But the case of an increasing number of synapses per neuron in large mammals can also be supported by the data, depending on how it is interpreted. What we can concluded is, that the average number of synapses per neuron in mammals is not higher than $2\text{-}3\cdot10^4$ and not lower than approximately 5000.

Table 2.5 contains a compilation of the anatomical data. This summary could have looked different, depending on what data that was considered relevant and accurate. We have followed two principles in the compilation of Table 2.5; Firstly, if data from many different studies agree, we have taken an average of these data. Secondly, in the case where only a few studies are available and the results in these studies are very different, we have relied on the more recent one.

In Table 2.6 we compute the neuron density per area and volume, based on the data in Table 2.5.

In Table 2.7 we estimate the total number of synapses by assuming a constant synaptic density across all species of $7 \cdot 10^8$ synapses·mm^{-3} and data from Table 2.5 on cortical surface area and thickness.

Three different estimates of the number of synapses per neuron are given in Table 2.8, in order to illustrate the range of possible values for this measure. The number of synapses per neuron varies with a factor 2-3 depending on how the total number of synapses has been estimated.

Table 2.5. The cortex data summarized for the five different mammals. † Computed, assuming $4.3 \cdot 10^4$ neurons·mm^{-3}. ‡ Computed, assuming $7.0 \cdot 10^4$ neurons·mm^{-3}. (These data are taken from Figure 2.2)

	Human	**Macaque**	**Cat**	**Rat**	**Mouse**
Gray Matter (cm^3)	490	55	15	0.4	0.1
White Matter (cm^3)	460	21	5	0.06	0.015
Cortex Area (mm^2)	$2.2 \cdot 10^5$	$2.5 \cdot 10^4$	$8.3 \cdot 10^3$	$6 \cdot 10^2$	$2 \cdot 10^2$
Thickness (mm)	2.7	2.1	1.7	1.2	0.8
Neurons	$2.0 \cdot 10^{10}$	$2.0 \cdot 10^9$	$6.1 \cdot 10^8$ †	$5.0 \cdot 10^7$ ‡	$1.6 \cdot 10^7$
Synapses	$1.5 \cdot 10^{14}$	$2.2 \cdot 10^{13}$	-	-	$1 \cdot 10^{11}$

Table 2.6. The density of neurons per area and volume. The data are computed from data in Table 2.5. Data marked with † were assumed in Table 2.5.

	Human	**Macaque**	**Cat**	**Rat**	**Mouse**
Neurons (mm^{-2})	$9.1 \cdot 10^4$	$8.0 \cdot 10^4$	$7.4 \cdot 10^4$	$8.3 \cdot 10^4$	$8.0 \cdot 10^4$
Neurons (mm^{-3})	$3.4 \cdot 10^4$	$3.8 \cdot 10^4$	$4.3 \cdot 10^4$ †	$7.0 \cdot 10^4$ †	$1.0 \cdot 10^5$

Table 2.7. The total number of synapses in cortex. These values were computed under the assumption that the synaptic density for all species is $7 \cdot 10^8$ synapses·mm^{-3}.

	Human	**Macaque**	**Cat**	**Rat**	**Mouse**
Synapses	$4.2 \cdot 10^{14}$	$3.7 \cdot 10^{13}$	$9.9 \cdot 10^{12}$	$5.0 \cdot 10^{11}$	$1.1 \cdot 10^{11}$

Table 2.8. The number of synapses per neuron. In the first row, without index, the data are taken from the literature. In second row the values are computed from the synapse data in Table 2.5. In the last row the values are computed based on the values given in Table 2.7.

	Human	**Macaque**	**Cat**	**Rat**	**Mouse**
Synapses / Neuron	7500	3900	7000	9000	7800
Synapses / Neuron	7500	11000	-	-	6300
Synapses / Neuron	21000	18400	16200	10100	7000

2.8.7 Analysis of Anatomical Data

In this section we present a summary and analysis of the anatomical data, on neuron and synapse numbers and cortical modularization. Here, we disregard species and area differences in the cortical structure. We compute the relevant figures for an implementation of the cortical model that we present in the next chapter.

If we assume that the average minicolumn is composed of 100 neurons, we know the total number of neurons, we can calculate the number of minicolumns (Table 2.9). The area density of neurons is roughly constant, 10^5 neurons·mm^{-2}, and therefore the average minicolumn diameter is about 40 µm. This diameter fits the figures found in the literature, which are based on direct measurements of the minicolumn's diameter.

There is a wide variety of shapes and sizes of hypercolumns. Based on the literature we assume that the hypercolumn is a circular structure with a diameter of about 400 µm. In this area it is possible to fit about 100 minicolumns with a diameter of 40 µm. This is also consistent with the figures found in the literature.

Table 2.9. The number of mini- and hypercolumns calculated for a number of mammals.

	Human	**Macaque**	**Cat**	**Rat**	**Mouse**
Minicolumns	$2.0 \cdot 10^8$	$2.0 \cdot 10^7$	$6.1 \cdot 10^6$	$5.0 \cdot 10^5$	$1.6 \cdot 10^5$
Hypercolumns	$2.0 \cdot 10^6$	$2.0 \cdot 10^5$	$6.1 \cdot 10^4$	$5.0 \cdot 10^3$	$1.6 \cdot 10^3$

A natural question, following the results presented above, is how many connections there exists between minicolumns. This can also be formulated as the degree of connectivity, i.e. the fraction of connections relative the case of full connectivity. Here, we assume that five synapses are required to establish a connection between two minicolumns, in one direction (appendix I). We do the calculation for two cases, as in Table 2.6; for a constant synapse density of $7 \cdot 10^8$ synapses·mm^{-3}, and for a constant number of 8000 synapses per neuron. The results are shown in Table 2.10.

In the case of a constant number of synapses per neuron the connectivity scales as $O(1/N)$.

Table 2.10. The number of minicolumn connections and mini-to-minicolumn connectivity in cortex. A minicolumn connection requires five synapses. † Here, we have assumed a constant number of 8000 synapses per neuron. ‡ Here, we have assumed a constant synapse density of $7 \cdot 10^8$ synapses·mm^{-3}.

	Human	**Macaque**	**Cat**	**Rat**	**Mouse**
Connections †	$3.2 \cdot 10^{13}$	$3.2 \cdot 10^{12}$	$9.8 \cdot 10^{11}$	$8.0 \cdot 10^{10}$	$2.6 \cdot 10^{10}$
Connectivity †	$8.0 \cdot 10^{-4}$	$8.0 \cdot 10^{-3}$	$2.6 \cdot 10^{-2}$	$3.2 \cdot 10^{-1}$	$1.0 \cdot 10^{0}$
Connections ‡	$8.3 \cdot 10^{13}$	$7.4 \cdot 10^{12}$	$2.0 \cdot 10^{12}$	$1.0 \cdot 10^{11}$	$2.2 \cdot 10^{10}$
Connectivity ‡	$2.1 \cdot 10^{-3}$	$1.8 \cdot 10^{-2}$	$5.3 \cdot 10^{-2}$	$4.0 \cdot 10^{-1}$	$8.8 \cdot 10^{-1}$

3 An Abstract Model of Neocortex

In this chapter we introduce an abstract functional model of neocortex and its mapping onto the cellular structure of cortex. This model is based on the columnar organization of cortex, and in particular on the idea of a repeated duplication of the columnar micro-circuitry throughout the cortical sheet (Markram, Toledo-Rodriguez et al. 2004). We present a hierarchical setup for this model and we also present a number of quantitative properties about the model such as the number of connections per unit, connectivity, and size of the hypercolumn modules. Finally, we discuss computational aspects and characteristics of cortex and cortical neurons in the context of the proposed model.

The topics discussed in this chapter are relevant to, and taken from, appendices I, III, VI, VII, and VIII.

3.1 Allometric and Theoretical Studies of Cortex

Neocortex has been formed and optimized by evolution during million of years. Studying how cortex has changed during evolution and how it scales between large and small mammals can provide important clues to its design and provide information on what evolution has optimized. It can also provide guidance in the design process of a cortical inspired computing system.

This has led many researchers to search for allometric scaling laws in which properties of the brain are related to each other and also to different anatomical features, often by power laws (Sultan 2002). Studying cortex from an evolutionary perspective (Kaas 1987; Krubitzer 1995; Northcutt and Kaas 1995; Barton and Harvey 2000; Barton 2002) or from a developmental perspective (Kaas 1987) are closely related to allometric studies in that the observed relationships can be explained using the same variables. Typically used variables in these studies are body mass, volume of gray and white matter, cortical surface area, and number of neurons.

Knowledge about large scale cortical characteristics can also be gained in a more indirect way by setting up a model cortex and then simulate evolution by optimizing one or several variables in the model before studying the outcome (Murre and Sturdy 1995; Kaas 2000; Zhang and Sejnowski 2000; Changizi 2001; Karbowski 2003). Commonly assumed constraining factors are the wiring length, energy consumption, neuropil size, and conduction times. One of the difficulties with this approach is that these variables are all intimately related in the brain, since adjustments to one of them will always affect the others.

One of the most common assumption is that wiring minimization is the driving force behind cortical organization (Cherniak 1995; Cherniak, Mokhtarzada et al. 2004). In a study considering several different network topologies it has been concluded that an exterior-sphere organization, such as cortex has, minimizes the wiring length (Murre and Sturdy 1995) and conduction delays (Wen and Chklovskii 2005). Similarly, it has been suggested that the folding of cortex does not only allow for a larger cortical surface area to be fitted inside the skull, but that it also minimizes the wiring (Scannell 1997). The distance between two cortical locations is drastically reduced if they are situated on opposite sides of a gyri.

Furthermore, it has been shown that the cortical areas in the prefrontal cortex of macaque are optimally placed to minimize their interconnecting wiring (Klyachko and Stevens 2003).

In a theoretical study of the optimal proportion of wiring in the gray matter that minimizes the conduction delays, it has been found that the optimal volume fraction of wiring is 3/5 (Chklovskii, Schikorski et al. 2002). This is in close agreement with the actual figures found in e.g. mouse. This study did only consider the neurons in the gray matter and did not include blood vessels and glial cells.

3.1.1 Scaling of Neocortex

There are basically two ways in which the cortex can be enlarged (Kaas 2000); Firstly, increasing the size of the neurons, which in turn enables the neurons to have more synapses. Secondly, increasing the number of neurons and thereby increasing the number of synapses. The first alternative has problems with the passive cable properties in the dendrites and the conduction delays in the axons. The second alternative has a problem with the connectivity, but by organizing the neurons into modules (hypercolumns, cortical areas) it can be resolved (Braitenberg 2001; Changizi 2001; Karbowski 2003).

In a large brain, thicker axons are required to maintain the same conduction times as in a small brain between opposite sides of a hemisphere. In the mammalian brain this seems to be the case, although there are very few of these thicker axons (Harrison, Hof et al. 2002).

3.2 Hebbian Learning and Attractor States in Cortex

A long-standing hypothesis is that the computations in cortex are performed based on associations between neurons. This idea was already captured in the early work of Donald Hebb (Hebb 1949) who postulated that the synaptic strength between two neurons is governed by their activity. At the time he made this postulate the activity dependent mechanism, which altered the synaptic strength, was unknown. Today it is known as long-term potentiation (LTP) (Malenka and Nicoll 1999). The strengthening of a synaptic weight due to LTP occurs only minutes after the initial, simultaneous, activation of two connected neurons has occurred and it can persist for hours or more. In some cases the neurons do not need to be activated simultaneously in order to produce LTP. Sometimes it is sufficient that the presynaptic neuron first is activated and then, later (both on timescales of ms and hours), the postsynaptic neuron becomes activated and LTP is induced (Abraham and Bear 1996; Frey and Morris 1997). There does also exist long-term depression (LTD) (Bi and Poo 2001), which is the opposite of LTP. It occurs, for instance, when only one of two connected neurons fires. The balance between LTP and LTD can depend on the exact timing of single spikes as in spike timing dependent plasticity (STDP) (Bi and Poo 1998). There are two excellent reviews of Hebbian learning, one with a focus on the underlying biology (Bi and Poo 2001) and one with a mathematical focus (Gerstner and Kistler 2002).

Although the basic principle of learning is simple in Hebb’s theory, the actual learning rules governing the changes in the synapses’ strength can be very complex and depend on a multitude of factors such as previous inputs, external neuromodulators, and the exact timing of action potentials. Experiments on human subjects testing sensorimotor learning have shown that the underlying processing is based on priors and probability distributions as in Bayesian statistics (Körding and Wolpert 2004).

Further, Hebb proposed the idea of cell assemblies (Palm 1981b), i.e. a number of neurons that are active simultaneously and connected with positive connections. The cell assembly theory follows as a consequence of the Hebbian learning where the synaptic strength is increased between co-active units. In Hebb's theory several of these cell assemblies can be active at any one time. It has been suggested that Hebbian cell assemblies can be modeled by recurrent neural networks (Palm 1981b; Lansner and Fransén 1992; Amit, Fusi et al. 1997; Lansner, Fransén et al. 2003). A prototypical attractor memory like the recurrent Hopfield network can be seen as a mathematical instantiation of Hebb's conceptual theory of cell assemblies. Today, experimental support for the existence of the stable states of cortical activity postulated by the cell assembly theory has been found (Cossart, Aronov et al. 2003; Shu, Hasenstaub et al. 2003).

In this chapter we propose an abstract model of cortex that is based on correlation based learning, such as LTP and LTD, and the idea of cell assemblies. An important difference to previous work done on cell assemblies is that the functional unit not is the neuron but a group of about hundred neurons, i.e. a minicolumn.

3.3 Modeling Cortex

When choosing between a number of different models for a particular system Ockham's razor states that the simplest model is to prefer. This model will have an adequate complexity to faithfully characterize the behavior of the system, but at the same time be simple enough to have a high explanatory power. In the case of modeling the brain and neocortex in particular, it is still unknown what the important characteristics are that we want to model. Therefore, constructing models with different levels of complexity that model cortex at different levels of abstraction is the best approach to understanding how it works. The models with a high level of abstraction can be used to study global dynamics and information processing aspects of cortex while more detailed models can be used to study the details of local interactions between neurons. Although these complex, detailed, models do not have the explanatory power of the simpler, abstract, models they are still useful in several ways; e.g. effects of different parameters such as ion channels and signal substances can be studied explicitly which is difficult to do in vivo or in vitro experiments.

As stated, models of the brain can have various degrees of complexity (Sandberg 2003) and in the following we exemplify three such different levels that are used by cognitive psychologists and neuroscientists. The most detailed models are the *compartment models*, in which a neuron is modeled with a finite number of volumes called compartments. The dynamics of each volume is governed by e.g. the Hodgkin and Huxley isopotential equations. The models belonging to the second level of intermediate abstraction are the *connectionist*, neural network type of, models. In a neural network model each unit models a single neuron or a group of neurons. The most abstract models, the *phenomenological* models, are those used mainly in psychology. Phenomenological models usually consist of a small number of related variables that are used to model simple and general behaviors of an entire neural system, e.g. classical conditioning models (Johansson and Lansner 2002a).

Even though today's cluster computers provide a tremendous computational capacity, choosing an appropriate level of abstraction is very important and in particular for cortex because of its huge complexity. The abstract model that is presented in the next section is based on the columnar organization of cortex. We feel that this model have a suitable level of complexity in that it captures the essential cortical computations while still being simple enough for efficient large-scale implementation. When considering a model of the cortex it is important that it scales well in terms of performance, implementation, and communication. Many of the connectionist models of cortical functions used by cognitive psychologists implements neural networks that are trained with non-local error gradient descent algorithms, e.g. back-propagation (Rumelhart, Hinton et al. 1986). Cortex does obviously not employ this type of non-local algorithms. A model of the cortex should be able to harness the power of the parallel computational structure, which the cortex has. This means that the algorithm must be local in space and probably also local in time, which is fulfilled by the Hebbian type of learning-rules.

3.4 A Generic Model of Cortex

Here we present an abstract generic model of how the mini- and hypercolumns are hierarchically organized in cortex. In this model the minicolumns are the smallest functional units and they are organized into hypercolumns that regulate their activity. The hypercolumns are in turn organized into cortical areas that finally build up the cortex as shown in Figure 3.1. In the previous chapter we concluded that a minicolumn is built up by approximately 100 neurons and that roughly 100 minicolumns are organized into one hypercolumn. The numbers on hypercolumns per cortical area and cortical areas in Figure 3.1 are very uncertain. We have assumed that the human cortex is divided into approximately 100 areas (Kaas 1987), and based on this computed the number of hypercolumns per cortical area.

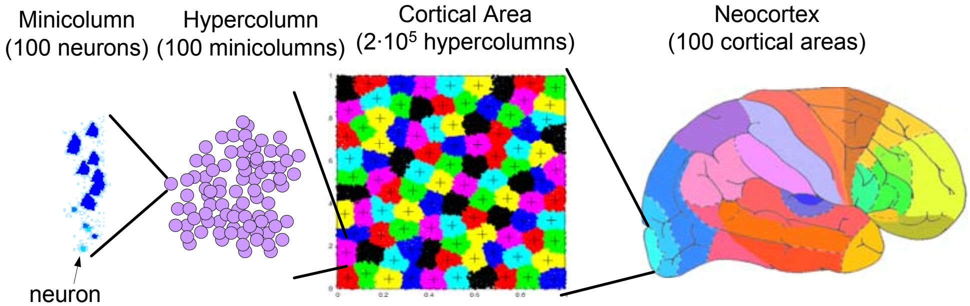


Figure 3.1. From neuron to neocortex, a schematic view of the cortical organization in human according to the abstract model proposed in this thesis.

Next we describe the hierarchically organization of hypercolumns and areas shown in Figure 3.2. All connections in this model are formed between minicolumns. The building block of this hierarchical structure is an area of highly interconnected hypercolumns (at the level of minicolumns) corresponding to a cortical area. These recurrently connected hypercolumns form an attractor network in the model. This recurrent connectivity is mainly formed by neurons in layers II/III and layers V/VI. Therefore, these four layers are taken together into one functional layer in the model that performs association (Figure 3.2).

The cortical areas of hypercolumns are connected by convergent forward projections and divergent backward projections as described in section 2.4.1. The forward projections originate on the presynaptic side from a local group of hypercolumns and converge to target a single or a few selected hypercolumns on the postsynaptic side. These projections originate in layers II/III on the presynaptic side and are received in layer IV on the postsynaptic side where minicolumns with specialized receptive fields are located. The backward projections originate from layers V/VI and target neurons over a wide area in layers II/III of the receiving cortical area.

The lower areas in the model drive the higher areas, which in term modulates the activity in the lower areas. The higher levels in the model feed eachother with input to layer IV, which make it possible for these areas to form holistic attractor states. In the brain these higher levels corresponds to the frontal and parietal cortex.

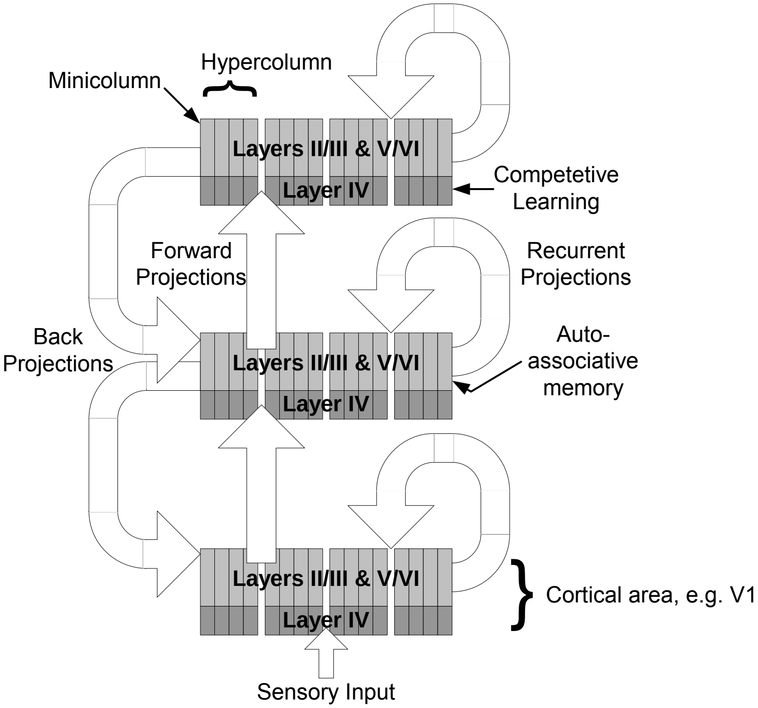


Figure 3.2. The hierarchical organization of hypercolumns and areas.

The role of the hypercolumns in the model is to regulate the activity in the minicolumns. The state of a minicolumn is described by its current activity and one or several memory traces of previous activations. In Figure 3.3 we show schematically how the minicolumns are grouped into hypercolumns. The minicolumns are submerged in a pool of inhibitory basket neurons that provides inhibition for all types of neurons in the hypercolumn (Cürüklü and Lansner 2002). The pyramidal neurons in a minicolumn have excitatory connections to the basket neurons and they receive inhibitory input from these. The reciprocal connection between all minicolumns and the basket neurons provides a normalizing soft winner-take-all (WTA) circuitry within the hypercolumn (Cürüklü and Lansner 2002). Inside each minicolumn there is another pool of small inhibitory interneurons, presumably of the double bouquet and bipolar types. These have a highly localized axonal arborization and provide inhibitory input to the local pyramidal neurons within one minicolumn. Corticocortical connections exist between minicolumns in different hypercolumns. If an incoming axon terminates on pyramidal neurons in the targeted minicolumn, the connection is excitatory, whereas if it terminates on inhibitory interneurons the connection is inhibitory. By adopting the minicolumn as the functional unit in our abstract model we get a network with both positive and negative couplings that still complies with Dale's law which states that a neuron can only provide either excitatory or inhibitory synapses, not both. Furthermore, in a network of minicolumns the connectivity is much higher than in a network of single neurons. This comes at the price of a smaller network, but on the other hand a very sparsely connected network may be totally useless and thus not meaningful (O'Kane and Treves 1992b). It has been suggested that mapping several neurons onto a single unit helps to resolve the discrepancy found in spike rates between in vivo neurons and the units of an integrate-and-fire network (higher spike rates in IF networks) (Lücke and Malsburg 2004). A last argument for networks of minicolumns is redundancy; if a cell in the minicolumn unit dies, the unit as such still remains functional in the network.

A connection between two minicolumns can be supported by one or several synapses. Most of the synapses in a minicolumn are devoted to receiving input from distant minicolumns situated in other hypercolumns, i.e. most synapses are devoted to corticocortical connections (Palm 1982). This was also the case in the model by Franzén and Lansner (1998), where an estimated number of 98.7% of the synapses were devoted to corticocortical connections. This means that the normalization circuit of the hypercolumn uses only a small fraction of the total number of synapses.

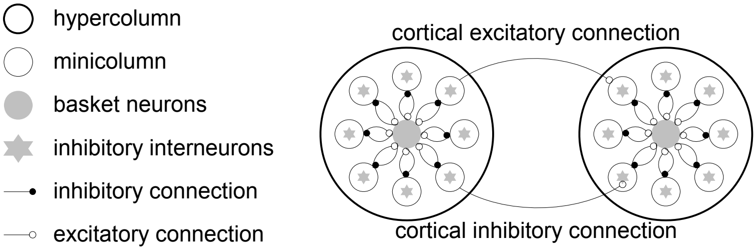


Figure 3.3. Two hypercolumns and their internal structure. Two corticocortical connections between minicolumns are shown, one that is excitatory and one that is inhibitory. The inhibitory effect comes from the excitation of inhibitory interneurons in the minicolumn receiving the afferent corticocortical axons. The normalization of activity in all minicolumns in one hypercolumn arises from the minicolumns reciprocal connections to the inhibitory basket neurons in the center of the hypercolumn.

3.4.1 Connectivity of the Minicolumns

The connectivity in the model at the level of minicolumns has been computed in two different ways, in appendix I it has been computed based on cortical anatomy and in appendix VIII it has been computed based on the neurons' physiology. The anatomical based method computes the number of synapses per minicolumn connection by considering how many other minicolumns a single minicolumn can potential contact and how many synapses that are devoted to these connections. The physiological based method considers the amount of synaptic input needed to a neuron and in turn also to a minicolumn in order to trigger activity. The result from both of these two methods is that a mini-to-minicolumn connection is supported by 5-15 synapses, and from this it is straight forward to compute the connectivity in the model since we know the total number of synapses.

In Table 3.1 the connectivity resulting from using the anatomically based method is presented. Here it is assumed that a mini-to-minicolumn connection is supported by 5 synapses. We point out that the number of connections scales linearly with the number of minicolumns, which means that the connectivity decreases as the network size is increased.

Table 3.1. Number of connections and connectivity in cortex. The connections are between minicolumns and the level of connectivity is computed as the fraction of full connectivity between the minicolumns. The majority of all synapses in a minicolumn are devoted to corticocortical connections, and each connection is on average supported by five synapses.

	Corticocortical Connections	Connectivity
Human	$2.4 \cdot 10^{13}$	$6 \cdot 10^{-4}$
Macaque	$3.6 \cdot 10^{12}$	$4 \cdot 10^{-3}$
Cat	$7.2 \cdot 10^{11}$	$2 \cdot 10^{-2}$
Rat	$6.0 \cdot 10^{10}$	0.24
Mouse	$2.4 \cdot 10^{10}$	0.94

3.4.2 Scaling the Model

Why design a network that scales with an increasing number of fixed sized modules instead of a network that scales by increasing the size of the modules? Both in the case of the neocortex and that of an implementation in silicon of the hypercolumns, their limited size makes it possible to implement the local competition, or any other local operation, using fast and cheap interconnections between the units. In the case of the neocortex, unmyelinated axons can be used which occupies a much smaller volume than myelinated ones. An implementation in silicon benefits from a local structure in that less space is used by transmission lines and a high clock frequency can be used. More generally, building a system by duplicating a building-block with fixed structure is easier than building a system where the structure of the building-blocks depends on the over all design of the entire system. That the growth of neocortex adhere to this design philosophy is evident from studies of ontogenetic columns (Rakic 1995).

3.5 Related Work

The generic model of cortex builds on cortically global interactions between minicolumns, similar to those in the model by Franzén and Lansner (1998), and on cortically local interactions within a hypercolumn, similar to those in the models by Cürüklü and Lansner (2002) and Sandberg and Lansner (2002, 2003).

The model presented by Franzén and Lansner (1998) does not incorporate the notion of hypercolumns; it models a network of minicolumns. The minicolumn network is used to implement an autoassociative memory. The model is based on the idea that a unit in a neural network does not correspond to a single neuron but to a minicolumn. This view helps to explain several discrepancies between the wiring in neural network models of autoassociative memory and the wiring of neurons in cortex. For example, it explains the existence of both inhibitory and excitatory corticocortical connections, the sparse connectivity of long-range cortical networks, and certain aspects of the local cortical wiring. The model is intended to be a generic model of cortex. Each minicolumn is modeled by 12 pyramidal neurons and 3 inhibitory interneurons, all located in layers II/III. A compartment-model of the neurons is used, based on a Hodgkin-Huxley formalism. A total of 50 minicolumns and 750 neurons are simulated, and these are connected by ~9000 connections. The minicolumns are spatially distributed over an area of 7×7 mm. The synaptic strengths in the model are derived from a recurrent neural network with the same dimensions as the model, which is trained with random patterns. The model, with the weights set as in the neural network, is able to do pattern completion and noise reduction on the trained patterns. A pattern completion is accomplished in 30-60 ms (Fransén and Lansner 1998).

Lundqvist et al. have continued the work on the model presented by Fransén and Lasner and added hypercolumns to the network of minicolumns (Lundqvist, Rehn et al. 2006a; Lundqvist, Rehn et al. 2006b). A hypercolumn is composed of a number of minicolumns and the minicolumns activities are normalized within the hypercolumn via inhibition provided by a few basket neurons. This model has been shown to be capable of attractor dynamics and it exhibits gamma oscillations during UP states i.e. when a memory is activated. The largest instance of this model that has been simulated was comprised of 2401 hypercolumns with 100 minicolumns in each, and it had 8 million neurons and 4 billion synapses (Djurfeldt, Johansson et al. 2005). Currently, only layers II/III and some neurons in layer IV are modeled but work is underway to extend the model with a complete layer IV and also layers V/VI.

Cürüklü and Lansner have presented a model based on integrate and fire neurons (Cürüklü and Lansner 2002). It is intended to simulate a hypercolumn in the visual cortex of cat. The hypercolumn is composed of 17 minicolumns and a large basket neuron. Integrate-and-fire neurons are used and each minicolumn is simulated by 11 excitatory and 3 inhibitory neurons in layer IV, and 12 excitatory and 2 inhibitory neurons in layers II/III. The excitatory neurons in layer IV are connected to the large basket neuron, which in turn inhibits the excitatory neurons in layers II/III. This model successfully demonstrates orientation selectivity (WTA), contrast invariance, and response saturation.

The computational properties, and in particular the feature extraction capabilities, of a network of minicolumns organized into hypercolumns have been studied by Lücke and Malsburg (Lücke and Malsburg 2004). In their work, each minicolumn was modeled with a few integrate-and-fire neurons. This model, although having more free parameters than an equivalent neural network model, showed to have an almost equal capability to those in the task of feature discovery.

Sandberg and Lansner (2002, 2003) have presented a yet more abstract version of the model presented by Franzén and Lansner, and Cürüklü and Lansner. In their connectionist type of neural network model the hypercolumns implements a soft WTA and the model's function as an autoassociative memory is studied extensively. In this thesis we work with a refined version of this model and we also consider more complex systems of networks based on this model.

3.6 Information Characteristics of a Synapse

From a computational perspective a synapse can be characterized in two ways: First, how much information it is capable of relaying (London, Schreibman et al. 2002); Secondly, the resolution in synaptic strength.

The amount of information a synapse is capable of relaying depends both on the synapse and the information it is relaying. Transmission rates over a synapse on a photoreceptor cell have been measured to 55 bits·s^{-1} (Laughlin, Steveninck et al. 1998).

The resolution of synaptic strength can be assessed by studying the process of quantal neurotransmitter release in the synapse. Recent findings suggest that the release of quanta is an all-or-nothing process (Silver, Lübke et al. 2003) and that also the potentiation of a synapse is an all-or-nothing process (Petersen, Malenka et al. 1998). Thus it seems that a synapse can be in one out of a few possible states and that the synaptic efficacy is limited to a small number of different levels, which means that a single synapse should be represented by 1-3 bits (Montgomery and Madison 2004). Given that a connection between two minicolumns is supported by five to fifteen synapses, using 8 bits to represent such a connection in the model seems justified.

3.7 Time-scales of Cortical Operation

The primary time-scale when simulating a network of electrically active units is the time it takes for a unit to change its electrical potential. This time sets an upper bound for the resolution needed in an emulation of cortex. Another defining timescale is that of the connection plasticity between units, how rapidly the strength of a connection can change. If one thinks of a system implementing an attractor memory it is also interesting to know the time it takes for the system to go to a fix-point.

The initiations of excitatory and inhibitory postsynaptic currents are fast processes that occur on timescales of less than 1 ms (Koch 1999). These currents change the electrical potential of the neuron, which forms the basis of neural computations. Approximately 50% of the spikes generated by pyramidal neurons are grouped, i.e. the spikes occur in a burst. The mean spike interval in a burst is approximately 10 ms. A single spike generates a low probability of synaptic release, but a train of spikes gives a probability of synaptic release close to 1. The high synaptic release probability comes at the price of a lower temporal resolution (Koch 1999). We take this as an argument for replacing a burst of high frequency spikes with a single spike in a neural network.

Synaptic plasticity occurs over a broad spectrum of time scales (Shepherd and Koch 1998; Koch 1999). The fastest event is facilitation that occurs on time scales of 10-100 ms. The slowest events, long-term potentiation and depression, occur one timescales of hours, days, or longer.

Experiments performed on monkeys indicate that the temporal precision of neural activity, in response to a visual stimulus, is on the average 10 ms and sometimes as high as 2 ms (Bair and Koch 1996). This means that cortical neurons are able to reliably follow events in the external world with a resolution of approximately 2-10 ms (Koch 1999). Orientation tuning in the orientation columns located in the visual cortex is a process that is hypothesized to relay on intracortical interactions. Studies on macaque monkeys show that it takes 30-45 ms for the orientation tuning to develop (Ringach, Hawken et al. 1997). Measurements in cat V1 have revealed that the patterns of cortical activity change on timescales of 40-80 ms (Kenet, Bibitchkov et al. 2003). In this study the sampling frequency was ~100 Hz, meaning that changes in activity faster than 10 ms were not recorded and thus not considered relevant. The probable timescale for attractor dynamics, i.e. convergence to a fix-point, is ranging from 30-200 ms (Fransén and Lansner 1998; Rolls and Treves 1998).

In this thesis we will refer to real-time operation as an update time of 10 ms, i.e. an update frequency F of 100 Hz. Although the activity of a neuron can change in less than 10 ms, this time-step is small enough to enable a model to have functional dynamics on an accurate cortical time-scale. This time-scale has also been proposed by Rolls and Treves (1998) as appropriate for the connectionist type of cortical models.

3.8 Communication Between Cortical Neurons

Neurons transmit information by voltage spikes. If the exact timing of each spike does not carry any information a rate code can be used, i.e. a number of spikes in a window of time are replaced by a real-valued average. That this type of coding is used in cortex has been confirmed by electrophysiological experiments in the inferior temporal cortex (Rolls, Aggelopoulos et al. 2004). More elaborate forms of rate based codes have been found in the entorhinal cortex, where neurons can have persistent activity that can be regulated between ~8 different frequency levels (Egorov, Hamam et al. 2002). A rate based code does not necessarily need to be defined as an average over time, but can also be defined as an average over a population of neurons (Gerstner 1998). Further, it is interesting to speculate whether spike bursts have a computational implication, such as the average frequency is the strength of the signal, or if they merely are a neurophysiological artifact. Some studies have suggested that all types of computational tasks can be accomplished with units that only have binary activity (Rolls and Treves 1998; Rolls, Aggelopoulos et al. 2004). Given that this is the case, it is sufficient to have spiking units with binary activity in a model of cortex.

There is also compelling evidence for a very sparse activity in networks of cortical neurons. A single spike in humans costs 3.3 times more energy than one in rat, and maintaining the resting potential costs 2.6 times as much, per neuron in humans, than in rats (Lennie 2003). Despite this, the glucose metabolism is three times lower per area unit in human than in rat cortex. This means that the neuron activity in humans must be sparser than in rats. Using a distributed coding with many neurons that have low firing frequencies is much more energy efficient than using a compact code with few neurons that have high firing frequencies (Laughlin, Steveninck et al. 1998; Attwell and Laughlin 2001). Further support for sparse coding has been found in microelectrode recordings performed in the cortex of monkeys. These recordings have shown that there is a 1‰ probability that a neuron is active within a time window of 10 ms (Abeles, Vaadia et al. 1990).

4 Attractor Neural Networks

In this chapter we present attractor neural networks. This type of networks has been used as a first very rough approximation of neural circuits in parts of the brain that are hypothesized to hold memories such as the neocortex and hippocampus (Little and Shaw 1975; Palm 1980; Palm 1981b; Fransén and Lansner 1998; Rolls and Treves 1998; Lansner, Fransén et al. 2003). We discuss their characteristics and in particular their memory storage capacity. The chapter starts with a quick introduction to artificial neural networks and their connection to neuroscience.

This chapter gives the background underlying the work in all of the appendices.

4.1 Artificial Neural Networks and Connectionism

Information processing in the cortex differs considerable from that of von Neumann machines (ordinary processors). The information processing in the cortex is highly parallel and fault tolerant. The cortical computational machinery has the ability to adapt (learn), it is robust, and it can deal with noisy and inconsistent data. These are properties that are desirable also in engineered systems, and thus they have motivated the study of artificial neural networks (Hertz, Krogh et al. 1991; Harvey 1994; Haykin 1999). The basic idea of a neural network is to have a large number of simple processors, called *units*, and connect them extensively. The state of a unit is represented by an activity level and this state-information is exchanged with other units via weighted connections. The network is trained by presenting it with example inputs that result in adjustments of the connection weights. From a neuroscience perspective, neural networks offer a way to deduce the computational properties of cortex by matching its physical architecture with abstract functional models that compute along the same line as cortex, i.e. with a large number of highly interconnected processing units. This branch of science is called connectionism. Cognitive psychologists have adopted the connectionist approach but to neuroscientists it has had less appeal because of its detachment from the actual physiology, e.g. ion channels and synaptic signaling substances, of neurons and synapses.

Neural networks are used in a wide range of research fields; signal processing, artificial intelligence, machine learning, and neuroscience. There are also numerous applications in which neural networks have been successfully applied; speech recognition, protein structure analysis, robotics, game playing, and handwriting recognition. In every field of research and application, different network structures and algorithms are used. There are almost a countless number of different variations (Hertz, Krogh et al. 1991; Haykin 1999). For a comprehensive compilation of the neural network taxonomies and hierarchies, see Gurney (Gurney 1997). Three different types of basic network architectures are; the single-layer feed-forward, the multi-layer feed-forward, and the recurrent networks. The recurrent network is the most general and the other two can be seen as special cases of this architecture. In fact, it has been shown that a recurrent network with a sigmoid transfer function is Turing complete (Siegelmann and Sontag 1991; Siegelmann and Sontag 1995). A special type of recurrent networks are the attractor neural networks, which are used in tasks such as optimization, clustering, associative memory, pattern completion, and noise reduction. The work in this thesis is focused on attractor networks primarily used for implementing content addressable autoassociative memory.

4.2 Attractor Networks and Autoassociative Memory

Autoassociative memory is a *content addressable* memory that can be implemented by an attractor network. A stored pattern is recalled from a partial or noisy version of it, which is called retrieval *cue*. The process of storing a pattern is referred to as *encoding* or *learning* and recalling a pattern is referred to as *decoding* or *retrieval*. During retrieval the activity of the network is iterated until a stable equilibrium is found and this process is called *relaxation*.

An attractor network can be formulated as a system of bipolar stochastic units (Hopfield 1982) or as a dynamical system with continuous valued units (Cohen and Grossberg 1983; Hopfield 1984). In the former case an analogy can be made to the Ising-spin system that is used in physics to describe the interactions in a ferromagnet and for which a number of mathematical tools exists that can be used for analyzing the system (Hertz, Krogh et al. 1991). This analogy was first pointed out by Hopfield in his seminal paper from 1982.

In this thesis we are interested in attractor networks that can be used as an abstract model of the neocortex. This means that we are primarily interested in networks that have sparse activity, a local learning-rule, and are capable of incremental learning. A spatially local learning-rule only requires the pre- and post-synaptic activities to update the weight-matrix and therefore it enables efficient parallelization. A learning-rule can also be local in time, meaning that the training examples can be presented one at a time, and this gives an incremental learning-rule. Algorithms based on a Hebbian learning-rule are local in both time and space and are also considered to be biologically plausible.

When an attractor network is used as a memory there is no advantage in using continuous or graded activity instead of binary (spiking) activity (Jonsson 2003; Johansson and Lansner 2005). But if the network is used e.g. as a preprocessing stage it is often useful to get a continuous valued output.

4.2.1 Performance Measures

Attractor networks are typically evaluated based on their storage capacity. The storage capacity can be defined in a number of ways, and taken to mean either the number of stored patterns or the amount of information stored per connection. Depending on how the storage capacity is evaluated, other aspects of the memory such as the size and shape of the attractor basins can also be taken into account. For an attractor network used as an autoassociative memory, round and equally sized basins of attraction are optimal and the network's ability to suppress noise in a retrieval cue is closely related to their shape. Also related to the memory performance are the number of spurious attractors and the probability of retrieving a spurious memory. Furthermore, the convergence time for a successful retrieval is also an important aspect of the performance. We can thus conclude that a high storage capacity does not necessarily guarantee that an attractor network, used as an autoassociative memory, will perform well.

In tasks such as clustering it is not obvious how to measure the performance. In this case, the emergences of prototype attractors that are superpositions of stored patterns, i.e. spurious attractors, are a desired feature. Amari has discussed that these mixed states could be the basis for "concept formation" (Amari 1977).

4.2.2 Experimental Investigation of Storage Capacity

When testing the storage capacity experimentally, the retrieval cue is normally a partial or noisy version of the original, stored pattern. This means that not only the number of attractors is measured but also the extent of their basins of attraction and stability. If unaltered original patterns are used as retrieval cues, only the stability is tested. The pitfall of evaluating attractor networks with unaltered versions of the original stored patterns is that the memories corresponding to the stored patterns may be only marginally stable and thus it is better to use noisy versions of the prototype patterns as retrieval cues.

We measure the storage capacity as the number of unique patterns that can be stored and retrieved from the network. This is a measure that is easy to implement and compute. But there are also more elaborate ways in which the storage capacity can be measured, e.g. it can be measured as the ability of the network to correct the information in a retrieval cue (Schwenker, Sommer et al. 1996). The drawback with this more elaborate method is that it is hard to compute in an experimental setting.

There are two types of policies for deciding the fraction of stored patterns. Firstly, it can be determined as the fraction of patterns that overlap the stored prototypes to 100%. Secondly, it can be computed as the average overlap between all trained and retrieved patterns. Throughout this thesis we have used the former method.

4.3 Network Models

The two most commonly used learning-rules for attractor networks are the Hopfield (Hopfield 1982; Hopfield 1984) and Willshaw-Palm (Willshaw, Buneman et al. 1969; Palm 1980) rules. The basic difference between these two algorithms is that the Hopfield rule produces real-valued weights and the Willshaw-Palm rule has binary weights. Both of these two rules implements a variant of Hebbian learning (Hebb 1949); the Willshaw-Palm model implements the asymmetrical coincidence rule and the Hopfield model uses the symmetrical covariance rule. The asymmetrical coincidence rule works with binary patterns, $\xi \in \{0, 1\}$, and only active units are associated, while the symmetrical covariance rule works with both $\xi \in \{-1, 1\}$ and $\xi \in \{0, 1\}$, and both inactive and active units are associated.

For both models the computations can be divided into two parts; training and retrieval. In the training phase the weights, w_{ij}, are computed and in the retrieval phase the units' activities, o_i, are updated.

In the retrieval phase a process called relaxation is run in which the activities of the units are iteratively updated. The *support*, s_j, is computed as a weighted sum of the presynaptic units' activities and is used to update the *potential*, m_j, which represents the state of a unit. Based on the potential value an *activity*, o_j, is computed, which is conveyed to other units via the interconnections. These connections have *weights*, w_{ij}, which modulate the input (activity) from one unit to the other. The weights are grouped into a *weight-matrix*, $\mathbf{w}$. Figure 4.1 shows a schematic model of how these computations are performed. In most implementations of the Hopfield and Willshaw-Palm learning-rules, we have that $\tau_m \rightarrow 0$ so that the potential is instantaneously set to the current support, making the differential equation for the potential updating superfluous.

Here, both the Willshaw-Palm and Hopfield models are discussed in the absence of local competitive groups of units, i.e. hypercolumns. It is straightforward to use these learning-rules, without any modifications, in a network with hypercolumns.

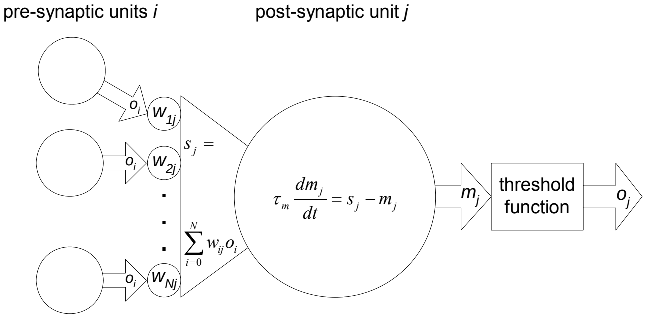


Figure 4.1. A schematic model of a unit in a neural network. The presynaptic inputs are weighted with connection weights, w_{ij}, to form the support, s_j, used to compute the potential, m_j, which is then finally used to compute the activity, o_j, i.e. the output. If the network is formulated as a dynamical system we have that τ_m>>0.

4.3.1 The Willshaw-Palm Model

This model owes it name to two different investigators. Willshaw et al. (1969) studied the associative capabilities of a two layers feed-forward network with binary weights, and Palm (1980) studied the autoassociative capabilities of a recurrent network with binary weights. The units in Willshaw-Palm networks are referred to as McCulloch and Pitts units for which $\tau_m \rightarrow 0$ in Figure 4.1 (McCulloch and Pitts 1943). They are defined by a binary output activity, $o_j \in \{0, 1\}$ and a non-linear threshold function (often a step function). The Willshaw-Palm model is always used with sparse patterns and it is well suited for digital hardware implementation due to its binary weights and activity. The symmetrical learning rule is described in eq. (4.1), where $\xi \in \{0, 1\}$.

$$w_{ij} = \bigvee^{Q} \xi_i \wedge \xi_j \tag{4.1}$$

4.3.2 The Hopfield Model

The patterns stored in the original Hopfield model are random, bipolar, $\xi \in \{-1, 1\}$, and unbiased $P(\xi_i=-1)=P(\xi_i=1)=0.5$. But it is also possible to store biased (sparse) patterns where, $\xi \in \{0, 1\}$ and $P(\xi_i=0) >> P(\xi_i=1)$. The learning-rule in eq. (4.2) describes how the weights are computed in the Hopfield model for sparse patterns, where N is the number of units, Q is the total number of patterns stored, and $a=P(\xi_i=1)$ is the activity level (Tsodyks and Feigelman 1988; Buhmann, Divko et al. 1989).

$$w_{ij} = \frac{1}{N} \sum_{\mu}^{Q} (\xi_i^{\mu} - a)(\xi_j^{\mu} - a) \tag{4.2}$$

The new activity, o_j, is typically computed with a sigmoid or a piece wise linear threshold function; $o_j \leftarrow f(m_j - \theta)$, where θ is the parameter setting the threshold. The dynamics of the Hopfield network can be both discrete (Hopfield 1982) ($\tau_m \rightarrow 0$ in Figure 4.1) and continuous (Hopfield 1984) (τ_m>>0 in Figure 4.1). In the discrete formulation, the states can be updated either asynchronously or synchronously and in the latter case the network is referred to as the Little model (Little and Shaw 1975). The long-time behavior of a network with asynchronous updating is identical to that of a network with synchronous updating (Amit, Gutfreund et al. 1985). The convergence time for a successful retrieval in a Hopfield network has been shown to be on the order of log$N$ (Tanaka and Yamada 1993). Further, in the limit of large networks and thus a low memory load, all spurious states correspond to well-defined mixtures of patterns (Amit, Gutfreund et al. 1985).

4.3.3 Other Learning-Rules

There exist a large number of other learning-rules besides the Hebbian type of learning-rules discussed in this thesis and these learning-rules typically give better results for non-sparse patterns than the Hebbian ones. Here we list a few of these learning-rules, the delta rule (Gardner, Wallace et al. 1989), the Widrow-Hoff rule (Widrow and Hoff 1960), the pseudo-inverse rule (Personnaz, Guyon et al. 1986; Diederich and Opper 1987; Kanter and Sompolinsky 1987; Krauth and Mezard 1987; Dotsenko, Yarunin et al. 1991), the eigenstructure rule (Yen and Michel 1992), and Storkey's rule (Storkey 1997; Storkey and Valabregue 1999). For a review on different learning algorithms for dense patterns see Abbott (Abbott 1990) and Davey et al. (Davey, Hunt et al. 2004).

4.4 Storage Capacity

The storage capacity of attractor networks is commonly measured for uniformly distributed random patterns where all units are equally likely to be active. The main reason being that attractor networks are good at storing this type of patterns because they have small mutual correlations.

The storage capacity can be computed for three different situations: Firstly, when all trained prototype patterns are stored without errors. Secondly, when most of the prototypes are stored without errors. Thirdly, when most of the prototypes are stored but with errors. For the Hopfield model, used with dense patterns, the storage capacity is proportional to $N/(4\log N)$ in the first case and $N/(2\log N)$ in the second case (McEliece, Posner et al. 1987). In the last case the storage capacity is expressed with the well-known formula $0.138N$ (Hertz, Krogh et al. 1991). In most papers on the storage capacity of attractor networks the last of the three definitions of storage capacity is used. For sparsely coded memories it can be shown that the first two definitions coincide with the third one asymptotically (Okada 1996).

Here we point out that most of the theoretical results on storage capacity are computed in the limit where $N\rightarrow\infty$. These results can therefore differ from experimental results achieved for finite sized systems.

The storage capacity of an attractor network can be studied either by a signal-to-noise (S/N) analysis or by studying the system's equilibrium equations (Okada 1996). The latter is achieved with a mean-field approximation in which the stochastic system is formulated with continuous-valued variables for which the equilibrium properties are studied. In the case of a low memory load the analysis is not too complicated, but in the case of a high memory load it becomes more complicated and a technique called replica theory is used (Hertz, Krogh et al. 1991). The basic S/N analysis traces back to work done by Amari (Amari 1977), but more elaborate forms exist such as SCSNA (self-consistent S/N analysis) (Shiino and Fukai 1993; Okada 1996). In this thesis we only use simple, single step, S/N analysis to investigate the storage capacity of attractor networks. This type of analysis does not capture the iterative nature of retrieval in attractor networks, but it is simple and generally gives fairly accurate results in case of sparse coding (Frolov, Husek et al. 1997).

Sparsely coded attractor networks have a very large storage capacity in terms of patterns (Willshaw, Buneman et al. 1969; Palm 1980; Tsodyks and Feigelman 1988; Buhmann, Divko et al. 1989; Golomb, Rubin et al. 1990; Meunier, Yanai et al. 1991; Okada 1996), which also increases with increased sparseness as $1/a\log a$ (Gardner 1987a; Gardner 1988; Tsodyks and Feigelman 1988; Okada 1996). But as the sparseness, a, increases the information content of each pattern decreases. This means that the optimum storage capacity, in terms of stored information, is achieved before $a \rightarrow 0$.

In terms of bits per unique connection the maximal storage capacity is $1/(2\log 2) \approx 0.72$ (Gardner 1987a; Gardner 1988; Palm and Sommer 1992; Frolov, Husek et al. 1997). In the case of the Willshaw-Palm model, the maximal storage capacity per unique connection is $\log 2 \approx 0.69$ (Willshaw, Buneman et al. 1969; Golomb, Rubin et al. 1990; Nadal and Toulouse 1990; Schwenker, Sommer et al. 1996). These capacity values, 0.69 for the Willshaw-Palm model and 0.72 for the Hopfield model, apply directly to feed-forward associative networks, but in the case of recurrent networks with symmetric weight-matrices, these values are divided by two (Buhmann, Divko et al. 1989). This means that we no longer consider only unique weights. The maximal storage capacity of the Hopfield model, with a symmetric weight matrix, has been computed to 0.36 bits per connection (Palm and Sommer 1992; Frolov, Kartashov et al. 1995b) and 0.38 (Buhmann, Divko et al. 1989). The maximal storage capacity of the Willshaw-Palm model, with a symmetric weight matrix, has been computed to 0.34 bits per connection (Palm and Sommer 1992; Frolov, Kartashov et al. 1995b).

If we want a memory with a vanishing error probability of the stored patterns, the storage capacities are again reduced with a factor ½ to $1/(4\log 2) \approx 0.18$ for the Hopfield model and $\log 2/2 \approx 0.17$ for the Willshaw-Palm model (Amit, Gutfreund et al. 1987; Palm and Sommer 1992; Frolov, Kartashov et al. 1995b; Frolov, Kartashov et al. 1995a; Schwenker, Sommer et al. 1996). These levels of storage capacity have been established by both single-step analysis and experiments. Furthermore, it is concluded that iterative retrieval does not improve the storage capacity over single step retrieval in the limit where $N \rightarrow \infty$ and $a \rightarrow 0$ (Schwenker, Sommer et al. 1996). Experiments have shown that for finite-sized networks the Willshaw-Palm model performs better than the Hopfield model and iterative retrieval is better than single step retrieval (Frolov, Kartashov et al. 1995b; Schwenker, Sommer et al. 1996).

4.5 Sparse Coding

Sparse coding is defined as the case where the number of active units in a memory-state or pattern is much smaller than $0.5N$. For the Willshaw network the optimal number of active units is $a = \log_2 N$ (Willshaw, Buneman et al. 1969), but for the Hopfield network finding the optimal level of activity is more complicated (Tsodyks and Feigelman 1988; Buhmann, Divko et al. 1989). For a review on associative memory and sparse coding see Okada (Okada 1996). Sparse patterns are referred to as biased or correlated patterns due to the unequal amounts of ones and zeros. In the case of sparse coding a Hebbian learning-rule is optimal (Hertz, Krogh et al. 1991; Sommer and Dayan 1998). The information content of a bipolar sparse pattern is computed as (Shannon 1948); $N(a\log_2 a + (1-a)\log_2(1-a))$, where $a = \mathrm{P}(\xi_i = 1)$.

4.6 Palimpsest Memory

An attractor network that forgets old patterns (memories) in favor of new patterns is said to be palimpsest (Nadal, Toulouse et al. 1986). Palimpsest attractor networks do not suffer from *catastrophic forgetting* due to overloading of memories as opposed to the standard Willshaw-Palm and Hopfield models. Catastrophic forgetting means that none of the stored patterns can be recalled from the memory and it occurs when too many patterns have been encoded.

The two best-known extensions of the Hopfield model to make it palimpsest are marginalist learning (Nadal, Toulouse et al. 1986) and learning within bounds (Hopfield 1982; Parisi 1986b), but there do also exist other palimpsest learning rules (Geszti and Pazmandi 1987; Shinomoto 1987; Christos 1996; Hemmen 1997; Robins and McCallum 1999). Marginalist learning works by imprinting new memories with an ever-increasing strength. Learning within bounds achieves the palimpsest property by introducing a limit to the weight-values (clipped weights). How to set this weight limit has been studied by Bonnaz (Bonnaz 1997). All of the above mentioned learning rules are analyzed with dense patterns (Mézard, Nadal et al. 1986; Nadal, Toulouse et al. 1986), but these learning rules can also be used for sparse patterns.

In the case of sparse coding, it has been shown that the information stored in a Hopfield network after training infinitely many patterns is $1/(\pi\log 2)\approx 0.459$ (Nadal and Toulouse 1990) with the standard Hebbian learning-rule. This means that in the case of sparse coding, a complete catastrophic forgetting does not occur. But the patterns retrieved from an overloaded memory have lots of errors and therefore it is advantageous to use a palimpsest learning-rule even though the catastrophic forgetting that would otherwise occur would not completely clean the network from previously stored information.

How a palimpsest memory can be implemented with a Hopfield type of network that is subject to constraints such as limited resolution in synaptic state variables has also been studied (Amit and Fusi 1992). Two reasons that make it interesting to study networks of this type that have constraints are that they more accurately match biology and that they are more amenable for realization in electronic hardware.

For the Willshaw-Palm model the synaptic coupling efficiencies are limited to two different values and for the synaptic couplings existing in the brain it is reasonable to assume that the coupling efficiencies also are limited to a few different values. It has been shown that a Willshaw-Palm network both can be palimpsest and achieve the optimal storage capacity of $(N/\log N)^2$, given that the synaptic potentiation and decay occur according to a certain type of learning rule and at a particular, low, rate (Amit and Fusi 1994).

Here we also note that the Willshaw-Palm model is well suited for hardware implementation because of its binary coupling efficiencies, but for this to be true it is also required that the updating of synaptic efficiencies does not require random numbers to be generated at each coupling for each update. Fortunately, if the input is stochastic, e.g. a Poisson distributed spike train, we do not need random numbers to be generated at each coupling (Brunel, Carusi et al. 1998; Fusi, Annunziato et al. 2000; Fusi 2001; Fusi 2002; Fusi and Senn 2006). This strategy typically requires that each coupling has an internal high resolution state variable that is deterministically updated by the incoming spike trains. But a more biologically realistic model that only requires a very limited number of unique states for the internal synaptic variables has also been devised (Fusi, Drew et al. 2005).

4.7 Spurious Attractors

Attractor networks typically have spurious attractors that do not correspond to any of the learnt patterns. In autoassociative memory and optimization tasks, the spurious attractors are considered bad, but in tasks such as clustering or a hierarchical memory system they are actually desired (Toya, Fukushima et al. 2000).

There are three types of spurious states in a Hopfield network (Hertz, Krogh et al. 1991); reversed states, mixed states, and spin glass states. In the case of a low memory load, only the first two types of spurious states are present. By setting the temperature, or noise level, to an appropriate value the mixed states can be avoided.

A central problem is that of distinguishing between attractors and spurious attractors (Robins and McCallum 2004). An obvious way of doing this is to classify a fix-point as spurious or not based on its energy, but this approach has not been very successful. Another solution is to study the variance of the support values for each unit in a fix-point. It has been proposed that the variances of support values are larger for spurious attractors (Robins and McCallum 2004).

The number of spurious attractors can be reduced in a number of ways. One approach is to unlearn the spurious attractors (Christos 1996; Hemmen 1997). A similar approach is pseudorehearsal (Geszti and Pazmandi 1987; Robins and McCallum 1999), where the desired memories are repeatedly trained. Another way of reducing the number of spurious attractors is to use an asymmetric weight-matrix (Parisi 1986a; Athithan and Dasgupta 1997).

4.8 Diluted Weight-Matrices

A diluted weight-matrix has less then N^2-N weights, which is the number of connections in a fully connected network without diagonal weights. If the elements are removed at random, the matrix will become asymmetric, which affects the fix-points dynamics. In this case convergence to a fix-point is not guaranteed and the network can potentially go into limit cycles (Hertz, Krogh et al. 1991; Sandh 2004). Fortunately, for networks with only slightly asymmetric weight-matrices, which are produced when a symmetric learning-rule operates on a randomly diluted connection matrix, this happens very rarely (Hertz, Krogh et al. 1991). For a network with a randomly diluted weight matrix, the variance of the synaptic input increases with decreasing levels of connectivity, which means that the storage efficiency in terms of bits per unique connection declines (Maravall 1999).

It has been shown that the weight-matrix of an attractor network can be heavily diluted and still retain a high storage capacity. In an investigation of the storage capacity of a Willshaw-Palm network, it has been found that even when 90% of the weights are removed (symmetrically) the storage capacity only drops from 0.34 to 0.26 bits per connection (Bosch and Kurfess 1998). The value of 0.26 bits per connection has also been reported by Palm (Palm 1980; Palm 1981a) and a similar value, 0.24 bits per connection, has also been given (Frolov, Kartashov et al. 1995a).

Interestingly, when the random connectivity is very sparse, i.e. the number of weights is less than $N\log N$, and the level of activity is low, a recurrent attractor network starts to behave as an associative feed-forward network (Treves 1991).

4.9 Threshold Control

The function that computes the activity from the potential is called the *threshold function*, which is also commonly referred to as the gain-, squashing-, activation-, or transfer-function in the literature. If the units have individually set thresholds there is the problem of balancing the overall activity level, it is prone to either grow without bounds until all units are active or die out (Frolov and Murav'ev 1993). If a local step-function is used the problem boils down to setting the value of a threshold variable, θ. In the limit of very large systems, i.e. $N \rightarrow \infty$, the threshold becomes less relevant (Dayan and Willshaw 1991). If the weight-matrix is diluted or if the activity level varies the problem of setting the threshold variable increases (Buckingham and Willshaw 1992; Frolov and Murav'ev 1993; Maravall 1999). The naïve threshold strategy is to set the threshold variable for each unit to a fixed value, but generally this gives poor performance (Buckingham and Willshaw 1993). More sophisticated threshold strategies use global information about unit usage and the global activity level to achieve better results. One such approach is the *k*-WTA (*k*-winner-take-all) strategy (Graham and Willshaw 1995), which at each time-step selects the *k* units with the largest support values to be active. Many other threshold strategies have also been proposed (Buckingham and Willshaw 1993; Hirase and Recce 1996; Schwenker, Sommer et al. 1996; Dominguez and Bollé 1998; Sommer and Dayan 1998). An optimal threshold strategy for the Willshaw-Palm network is proposed by (Buckingham and Willshaw 1993). This strategy sets the thresholds individually for each unit so that the output error is minimized. It, however, requires extensive information about input activity, density of modified synapses, unit usage, and the amount of noise in the retrieval cues. It has been shown (Graham and Willshaw 1995; Graham and Willshaw 1997) that a *k*-WTA with normalization due to input activity and unit usage performs almost as well as the optimal threshold strategy.

From the perspective of parallel implementation, threshold control strategies that only require local information are preferred. Threshold strategies such as the *k*-WTA and the optimal threshold strategy proposed by Buckingham and Willshaw are therefore not suitable for parallel implementation. In this thesis we let the hypercolumn regulate the activity level, which makes moderate use of global information and it is therefore well suited for parallel implementation (next chapter).

By using a non-monotonic transfer function attractor networks with capabilities exceeding those of the typical Hopfield model can be constructed (Morita 1993; Shiino and Fukai 1993; Yanai and Amari 1996).

4.10 Higher-Order Networks

Networks that compute their units' supports from a combination of additive and multiplicative terms are referred to as a higher-order networks or networks with higher-order units (Grossberg 1988; Karayiannis and Venetsanopoulos 1990; Schmitt 2001). Two examples are sigma-pi networks and networks with product units. *Sigma-pi* units have an input formed by a sum of *monomials* and *product* units are defined by having an input formed by a product of weights and activities. Another category of higher order networks are those where units, and groups of units, are connected with multiple connections (Abbott and Arian 1987; Gardner 1987b). Higher-order networks can also be built with multi-layered networks or multiple networks of summing units given that they have a non-linear threshold function. Such architectures have been proposed by several authors (O'Kane and

Treves 1992a; O'Kane and Sherrington 1993; Levy, Horn et al. 1999; Renart, Parga et al. 1999; Johansson 2001b). The multi-layer instance of the cortical model adheres to this class of networks.

There are also networks in which the activity is represented in a more complex way than in the standard Hopfield network. Two examples are Q-state networks (Cook 1989; Yedidia 1989; Rieger 1990; Mertens 1991; Zurada, Cloete et al. 1996) and Q-state networks with complex valued weights (Noest 1989; Muezzinoglu, Guzelis et al. 2003).

5 Attractor Networks with Local Competition

This chapter focuses on a special type of modular attractor networks with local competition. These networks' units are organized into non-overlapping groups called hypercolumns, in which they compete for activation. The foremost purpose, and function, of the hypercolumn is to regulate the activity level. In Figure 5.1 a network with *H*=3 hypercolumns and *U*=3 is shown. It has a total number of *N*=9 units. There are no recurrent connections within a hypercolumn.

This chapter, similar to chapter 4, discusses issues underpinning the work in all of the appendices, in particular it presents the results from appendices II and IV.

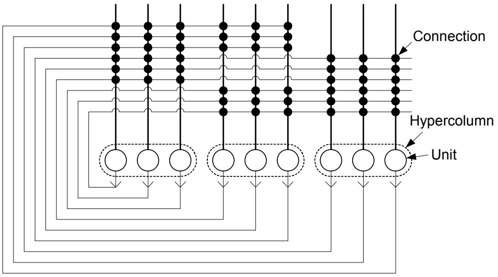


Figure 5.1. A modular network with a total of 9 units divided into 3 equally sized hypercolumns. Within each hypercolumn the units compete for activation. The network is fully connected and there are no connections between the units within a hypercolumn.

5.1 Potts Neural Networks

In Potts neural networks the units are multi-valued and the analog spin-system is the Potts-spin model (Kanter 1988). It has been pointed out that the multi-valued coding in Potts networks is suitable for representing many different types of data (Nadal and Rau 1991; Vogt and Zippelius 1992; Wuagh and Westervelt 1993a). Potts networks have been used as both attractor and associative memories and they have been investigated by several authors (Kanter 1988; Bollé and Mallezie 1989; Bollé, Dupont et al. 1991; Nadal and Rau 1991; Shim, Kim et al. 1991; Bollé, Dupont et al. 1992b; Bollé, Dupont et al. 1992a; Bollé, Dupont et al. 1992c; Ferrari, Martínez et al. 1992; Gayrard 1992; Shim, Kim et al. 1992; Vogt and Zippelius 1992; Wuagh and Westervelt 1993a; Wuagh and Westervelt 1993b; Kartashov, Frolov et al. 1997). Potts networks have also been applied to optimization problems (Ishii and Sato 1997; Häkkinen, Lagerholm et al. 1998; Häkkinen, Lagerholm et al. 2000; Lagerholm, Peterson et al. 2000; Tang, Tan et al. 2004), independent component analysis (Wu and Chiu 2001), and self-organization – clustering tasks (Liou and Wu 1996; Bengtsson and Schubert 2001). Similar to the Potts network, there are multi network systems that capture the idea of using locally competing binary units to represent multi-valued features in an associative memory (Wersing, Steil et al. 2001). In this thesis we use networks in which binary or real-valued units are grouped together into hypercolumns, in which they compete for activation (Figure 5.1). A unit in a Potts-network corresponds to a hypercolumn in these networks. In the following H is the number of Potts units in a network and U is the number of different values that they can take on.

In a Potts network there can be only one pattern active at any one time as opposed to the standard Hopfield model, where several patterns can be active simultaneously (Kanter 1988). This follows as a consequence of that only a single state can be active in a Potts-unit at any one moment. It has been found that the number of spurious states is smaller in a Potts network than in a corresponding Hopfield network (Bollé, Dupont et al. 1991; Wuagh and Westervelt 1993b). It has also been concluded that the storage capacity of a Willshaw-Palm network equals that of a Potts network used with binary weights in the limit of $N\to\infty$ (Kartashov, Frolov et al. 1997). Kartashov et al. suggest that the optimal partitioning of a hypercolumnar network is given by $H=\log_2 U$. If we instead use the prescribed activity level for Willshaw-Palm networks we get a somewhat larger number of hypercolumns $H=\log_2 HU=\log_2 N$.

The storage capacity of the Potts network has been investigated by a large number of authors. Shim et al. (1991) studied a Potts network with binary connection weights. Bollé et al. (1989) have studied a Potts network with asymmetric weights. A Potts network formulated as a dynamical system has also been investigated (Wuagh and Westervelt 1993a; Wuagh and Westervelt 1993b). Using the Potts network with biased patterns where the units in the hypercolumns have different probabilities of being active has been studied by Bollé et al. (Bollé, Dupont et al. 1991; Bollé, Dupont et al. 1992b; Bollé, Dupont et al. 1992a) and by Shim et al. (Shim, Kim et al. 1992). It has also been pointed out that the attractors can have a hierarchical relationship (Vogt and Zippelius 1992). Kanter (1988) proposed an approximation formula of the Potts-network's storage capacity; $0.138N(U-1)/2$, and that it scales as $N(U-1)$. The storage capacity of a feed-forward Potts-network was investigated with Gardner's technique by Nadal and Rau (Nadal and Rau 1991). In their analysis, stability of all stored patterns was required and they found that the storage capacity scales as $H(U-1)$. Studying a feed-forward network, Shim et al. (1992) computed the storage capacity of patterns to $NU^{0.85}$ in the limit $H\to\infty$ and $U\to\infty$.

5.2 BCPNN

The Bayesian Confidence Propagating Neural Network (BCPNN) was originally developed as an attractor network in analogy with the Hopfield model (Lansner and Ekeberg 1989). The BCPNN is derived from Bayes' formula, lending ideas from the naïve Bayesian classifier (Mitchell 1997), which makes it appealing from a statistical viewpoint. In this derivation, the hypercolumn structure of the network naturally arises, as it is necessary to represent mutually exclusive events. A similar learning-rule has been studied by Kononenko (Kononenko 1989).

The BCPNN has been used as a classifier in both single- and multiple-layer feed-forward and recurrent network architectures (Tråvén 1993; Levin 1995; Lansner and Holst 1996; Holst 1997; Orre 1998). It has been applied to character recognition (Tråvén 1993) and appendix VI; datamining (Orre 1998; Orre, Lansner et al. 2000); storing temporal sequences (Levin 1995; Orre 1998); hierarchical clustering of stored memories (Levin 1995; Eriksson and Lansner 2003; Orre, Bate et al. 2005) and appendix IV; reinforcement learning (Johansson and Lansner 2002b; Johansson, Raicevic et al. 2003); modeling of classical conditioning (Johansson and Lansner 2002a); and connectionist modeling of memory (Lansner and Ekeberg 1987; Sandberg, Lansner et al. 2000; Sandberg and Lansner 2001;

Sandberg and Lansner 2002; Sandberg, Lansner et al. 2002; Sandberg 2003; Sandberg, Tegnér et al. 2003).

First we present the updating of the units' activities and then the learning rule. For a detailed derivation of the algorithm see e.g. Holst (1997) and Sandberg (2003). The network can be used with either unary-coded activity (spiking activity), $O \in \{0, 1\}$, or real-valued activity, $o \in (0, 1)$. The network has N units grouped into H hypercolumns with U_h units in each. Here, h is the index of a particular hypercolumn and Q_h is the set of all units belonging to hypercolumn h. We also have that $Q_h \cap Q_k = \emptyset$, $\forall h,k{:}h \neq k$. The computations of the network can be divided into two parts; training and retrieval. In the training phase the weights, w_{ij}, and biases, β_j, are computed. In the retrieval phase the units' activities, o_i, are updated. The activity sum to 1 for each hypercolumn. Each hypercolumn is intended to represent a particular attribute, e.g. color, and each unit is intended to represent a value of an attribute, e.g. red. Next, we first present the retrieval phase and then the training phase.

The retrieval phase (Figure 5.2) is similar to that of the Willshaw-Palm and Hopfield models. The two differences being the columnar structure and the bias term, β_j. In the case of units with real-valued activity the transfer function is a softmax function and in the case of units with binary activity, the active unit in each hypercolumn is stochastically selected according to a distribution generated by the softmax function.

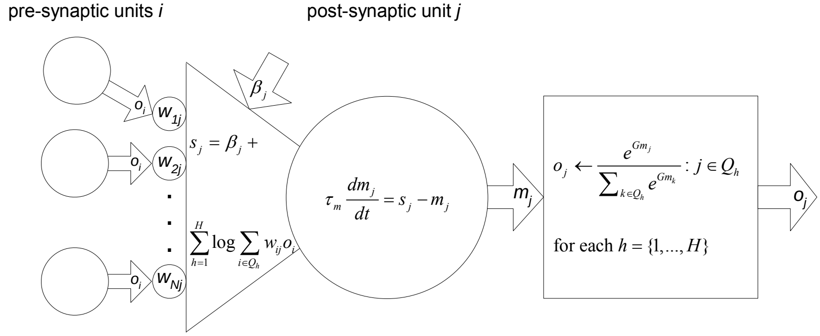


Figure 5.2. A schematic model of a unit in a BCPNN. The presynaptic inputs are weighted with the connection weights, w_{ij}, to form the support, s_j, used to compute the potential, m_j, which is then finally used to compute the activity, o_j, i.e. the output.

Here we present the learning phase in three different versions; Figure 5.3, Figure 5.4, and Figure 5.5. Here, S is the input and Z, E, and P are the state variables and each of the state variables are associated with a time constant τ that governs the speed by which it can change. Each unit has a bias, β_j, that is computed based on the probability, P_j, of that particular unit being active. The weights, w_{ij}, are computed based on the probabilities P_i of the presynaptic unit being active, P_j of the postsynaptic unit being active, and P_{ij} of that both the pre- and postsynaptic units being active simultaneously. If the sole goal is to implement an associative memory it is enough to have the P state variables as in Figure 5.3, Z and E variables are then superfluous. The Z variables are intended for synaptic spike integration and the E variables implements an eligibility trace. The purpose of the eligibility trace is to enable control of which correlations are stored in the memory variables P and it enables implementation of reinforcement learning (Johansson and Lansner 2002b; Johansson, Raicevic et al. 2003). With the full set of variables and equations it is possible to accurately model spike time dependent plasticity (STDP) data (Wahlgren and Lansner 2001).

Here, the state variables are updated by first order, ordinary, differential equations and take an input that is continuous in time. These differential equations can also be interpreted as exponentially weighted moving averages (EWMA), which are commonly used in statistical process control (Tseng, Yeh et al. 2003). The equations are also known as first order low-pass filters with infinite impulse response (Oppenheim and Willsky 1997). Further, the equations are similar to those of leaky integrators, which are widely used in implementations of both abstract integrate-and-fire (Jolivet, Lewis et al. 2004) and biophysically detailed (Koch 1999) models of neurons. The state variables are initialized as $P_i=1/U$ and $P_{ij}=1/U^2$, and similarly for Z and E. In an actual implementation it can be necessary to use a term λ_0 as in eq. (5.1) that protects from taking the logarithm of zero.

$$\tau_P \frac{dP_i(t)}{dt} = ((1-\lambda_0)S_i(t)+\lambda_0)-P_i(t) \tag{5.1}$$

There are no connections between the units of a hypercolumn. This means that H blocks of U^2 connections are deleted in the diagonal of the weight matrix and in the case of full connectivity a recurrent network has $N(N-U)$ connections.

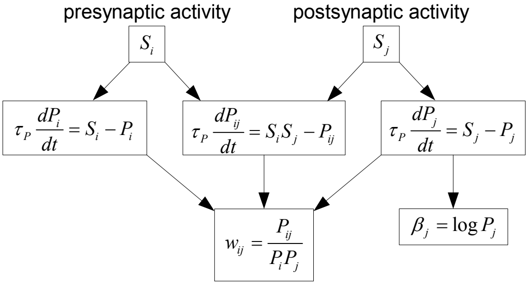


Figure 5.3. The minimal set of equations needed to implement a BCPNN of P-type.

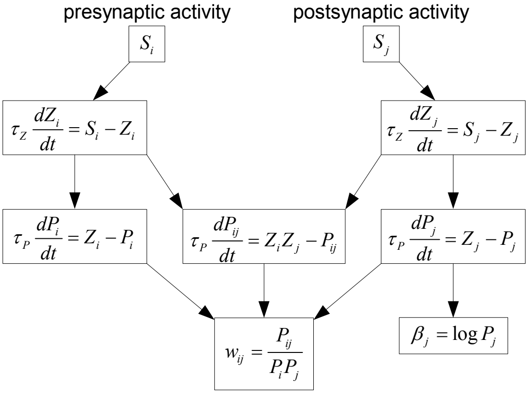


Figure 5.4. A Z-type BCPNN with an intermediate level of complexity.

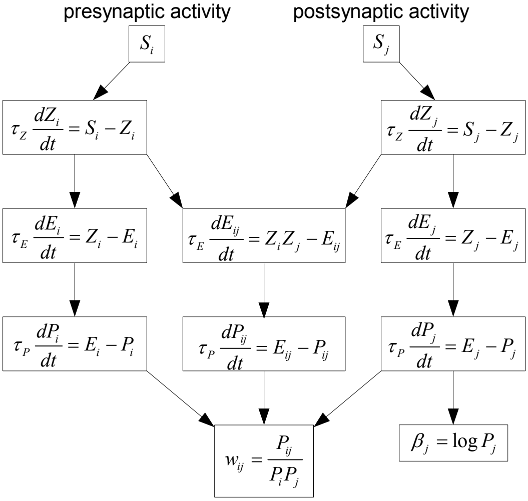


Figure 5.5. The equations of the E-type BCPNN, which is more complex that the other two versions.

5.2.1 Probabilities Estimated by Relative Frequency

When the relative frequency is used to estimate the probability values P_i and P_{ij}, it is assumed that there are a finite number of Q inputs $S\in(0,1)$. In previous work this way of estimating the probabilities is referred to as summing or counting (Holst 1997). A relevance signal $\kappa\in(0,\infty)$ regulates the relevance or strength of each input and κ is normally set to 1 for all inputs. In eq. (5.3) the activity of a *presynaptic* unit, *i*, is correlated with that of a *postsynaptic* unit, *j*. The sums in eq. (5.2)-(5.4) are used to calculate the probabilities; $P_i=c_i/C$ and $P_{ij}=c_{ij}/C$.

$$c_i = \sum_{t=1}^{Q} \kappa(t) S_i(t) \tag{5.2}$$

$$c_{ij} = \sum_{t=1}^{Q} \kappa(t) S_i(t) S_j(t) \tag{5.3}$$

$$C = \sum_{t=1}^{Q} \kappa(t) \tag{5.4}$$

When the β_i and w_{ij} values are computed (eq. (5.5) and (5.6)) the explicit probability estimates (P_i and P_{ij}) are actually not needed. Instead, the β_i and w_{ij} values are calculated directly from the counts, c_i and c_{ij}.

$$\beta_i = \begin{cases} \log 1/C^2 & c_i = 0 \\ \log c_i / C & \textit{otherwise} \end{cases} \tag{5.5}$$

$$w_{ij} = \begin{cases} 1 & c_i = 0 \vee c_j = 0 \\ 1/C & c_{ij} = 0 \\ \frac{c_{ij} C}{c_i c_j} & \textit{otherwise} \end{cases} \tag{5.6}$$

5.2.2 Probabilities Estimated by Exponentially Weighted Moving Averages

The probability values P_i and P_{ij}, and also all of the other state variables, can be updated by first order, ordinary, differential equations as;

$$\tau \frac{dp(t)}{dt} = \kappa(t)(s(t) - p(t)) \tag{5.7}$$

where κ is a relevance signal that is set to 1 in the following equations. Here, $p\in(0,1)$ is the continuously updated estimate, $s\in(0,1)$ is a time varying input, and $\tau\in[0,\infty)$ is a plasticity parameter that controls the speed of the dynamics. Solving eq. (5.7) numerically by Euler's method we get:

$$p_{t+1} = p_t + \frac{s_t - p_t}{\tau}\Delta t \tag{5.8}$$

Here, Δt is an arbitrarily chosen integration time constant that is set to 1 in the following. The recursion in eq. (5.8) can also be interpreted as an exponentially weighted moving averages (EWMA), that weights older values of the input s with decreasing strength (appendix V):

$$p_t = \frac{(\tau-1)^t}{\tau^t} p_0 + \sum_{k=0}^{t-1} \frac{(\tau-1)^{t-k-1}}{\tau^{t-k}} s_k \tag{5.9}$$

5.2.3 The m-Estimate

Here we show how the EWMA can be interpreted as a Bayesian m-estimate with a constant weighting of the input samples. The m-estimate is here derived using Bayesian statistics, which shows that updating the state variables in a BCPNN with EWMA:s fit well into the Bayesian framework from which the BCPNN was derived.

Accurate and reliable estimates of parameters are essential to a large number of algorithms, e.g. the performance of the naïve Baysian classifier is heavily dependent upon estimates of the conditional probabilities (Cestnik 1990; Mitchell 1997). It is often the case that parameters must be estimated from very few and noisy samples. A commonly used estimate under these circumstances, with good characteristics, is the m-estimate (Cestnik 1990; Mitchell 1997). The m-estimate can incorporate prior information into a parametric density estimation problem. The m-estimate is used in a diverse set of filtering applications, where it is efficiently used to remove impulse noise.

We start by considering the problem of estimating the probability p of getting heads instead of tails when flipping a coin, i.e. estimating a probability from binomial data. The following calculations are based on those in Gelman et al. and Holst (Holst 1997; Gelman, Carlin et al. 2004), but they are done in a way so that the connection to the m-estimate can be shown. The probability p is the model, *Model*, and when flipping the coin we count the number of heads, c_H, which is the data, *Data*. The probability of getting c_H heads out of $c=c_H+c_T$ tosses is

$$\Pr(Data \mid Model) = \Pr(c_H \mid p) = \binom{c}{c_H} p^{c_H}(1-p)^{c_T} \tag{5.10}$$

In classical statistics the best estimate of p is the relative frequency c_H/c; which is the maximum likelihood estimate. This estimate maximizes the probability, by adjusting the model *Model* (expressed by p), of getting the data *Data* (which we already have). This method of estimation runs into problems if there are few or no samples of the data. In situations with little or no data, a best guess is given by a Bayesian estimate that assumes a prior of p. Later, when more data is available the estimate is refined. In the case of Bayesian statistics we want to maximize Pr(*Model*|*Data*), i.e. the probability of the model *Model* given the collected data *Data*. By using Bayes formula we can write:

$$\Pr(Model \mid Data) \propto \Pr(Model)\Pr(Data \mid Model) \tag{5.11}$$

Here, Pr(*Model*) is called the prior probability and Pr(*Model*|*Data*) the posterior probability. A general prior probability density function is

$$\Pr(Model) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} p^{a-1}(1-p)^{b-1} \tag{5.12}$$

where Γ(x) is the gamma distribution and a and b are arbitrary parameters. With this prior we can write the posterior distribution as

$$\Pr(Model \mid Data) = \Pr(p \mid c_H, c_T) \propto p^{a-1}(1-p)^{b-1} p^{c_H}(1-p)^{c_T} \tag{5.13}$$

where we have dropped all terms that do not depend on p. The integral of (5.13) is a Beta distribution with the parameters $a+c_H$ and $b+c_T$:

$$\int_0^1 p^{a-1}(1-p)^{b-1} p^{c_H}(1-p)^{c_T}\,dp = \frac{\Gamma(a+c_H)\Gamma(b+c_T)}{\Gamma(a+b+c_H+c_T)} \tag{5.14}$$

The probability density function for the posterior distribution is computed as:

$$\Pr(p \mid c_H, c_T) = \frac{\Gamma(a+b+c_H+c_T)}{\Gamma(a+c_H)\Gamma(b+c_T)} p^{a-1}(1-p)^{b-1} p^{c_H}(1-p)^{c_T} \tag{5.15}$$

Next we calculate the expectation of p, E(p), which we use as an estimate of p:

$$\begin{aligned}
&\hat{p} = E(p) = \int p \Pr(p \mid c_H, c_T)\,dp = \\
&\frac{\Gamma(a+b+c_H+c_T)}{\Gamma(a+c_H)\Gamma(b+c_T)} \int_0^1 p^{a}(1-p)^{b-1} p^{c_H}(1-p)^{c_T}\,dp = \\
&\frac{\Gamma(a+c_H+1)\Gamma(a+b+c_H+c_T)}{\Gamma(a+b+c_H+c_T+1)\Gamma(a+c_H)} = \frac{a+c_H}{a+b+c_H+c_T} = \\
&\frac{c_H+a}{c+a+b}
\end{aligned} \tag{5.16}$$

This estimate of p minimizes the expected error in eq. (5.17), given that the cost function $L(\cdot)$ is the squared error.

$$\int L(p,\hat{p}) \Pr(p \mid c_H, c_T)\,dp \tag{5.17}$$

If we instead want to minimize the probability of giving the wrong answer, $L(\cdot)$ should be set constant and the relative frequency should be used as an estimate:

$$\hat{p} = \frac{c_H}{c} \tag{5.18}$$

If we set the parameters in eq. (5.16) to a=0 and b=0 the estimate is equal to that of relative frequency and if they are set to a=1 and b=1 (uniform prior) we get Laplace's law of succession:

$$\hat{p} = \frac{c_H+1}{c+2} \tag{5.19}$$

The estimate in (5.16) is called an m-estimate (Cestnik 1990) when the parameters in the a prior distribution are chosen as; $a=pm$, where p is the old estimate, and $b=m-a$.

$$\hat{p} = \frac{c_H + mp}{c + m} \tag{5.20}$$

Here, m is a parameter controlling how much weight that is given to the old estimate and it is referred to as the equivalent sample size. If we use the m-estimate to update our estimate p_t after collecting a single sample (c=1) denoted by s_t, we get a single step incremental m-estimate:

$$p_{t+1} = \frac{s_t + mp_t}{1+m} \overset{\tau=1+m}{\Rightarrow} p_{t+1} = \frac{s_t + (\tau-1)p_t}{\tau} \Rightarrow p_{t+1} = p_t + \frac{s_t - p_t}{\tau} \tag{5.21}$$

This expression is identical to that of the EWMA in eq. (5.8), for which the equivalent sample size is controlled by τ.

5.3 Energy Function

A characterizing feature of attractor networks with summing units is that they have a symmetric weight-matrix and this in turn is a prerequisite for formulating an energy function (Hertz, Krogh et al. 1991). The energy function is defined as a function of the network's state (activity) and structure (weights), which never increases in value as the activity is updated. The existence of an energy function makes it possible to interpret the dynamics of the attractor network in terms of an energy landscape, in which a memory corresponds to a *local minimum*. A local minimum is surrounded by a *basin of attraction*, which means that all of the states surrounding this minimum are attracted to it. Many of the theoretical results on attractor networks, such as the storage capacity, are often based on the energy function. The energy function is also used when solving optimization problems (Peterson and Söderberg 1989). The energy function is referred to as the Hamiltonian function in statistical mechanics, the Lyapunov function for dynamical systems, and objective function in the case of optimization (Hertz, Krogh et al. 1991).

The binary activities, O, of the units in a Hopfield network are asynchronously and stochastically updated according to eq. (5.22). Hopfield showed that this type of network, with a symmetric matrix $\boldsymbol{w}$, admits to the energy function in eq. (5.23).

$$O_j \leftarrow \text{sign}\left(\sum_{i=1}^{N} w_{ij} O_i\right) \tag{5.22}$$

$$E = -\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} O_i O_j \tag{5.23}$$

The dynamics of this type of stochastic system can be approximated by a set of mean-field equations, i.e. binary stochastic variables are replaced with average values. The dynamics of a network can then be described by a set of deterministic differential equations. An analysis of the system can then determine the speed of convergence to a fix-point and also its storage capacity in terms of attractor states.

Cohen and Grossberg showed (Cohen and Grossberg 1983; Grossberg 1988) that a nonlinear system of differential equations as that described in eq. (5.24) admits to an energy function, eq. (5.25), under three conditions: Firstly, the matrix $\boldsymbol{w}$ must be symmetric. Secondly, the nonlinear threshold function, $f(m_j)$, must be monotonously increasing, i.e. $f'(m_j) \geq 0$. Thirdly, $a_j(m_j) \geq 0$. Here, the state variables, m_j, are real-valued as opposed to the bipolar valued ones, O, in the Hopfield network.

$$\frac{dm_j}{dt} = a_j(m_j)\left(b_j(m_j) + \sum_{i=1}^{N} w_{ij} f(m_i)\right), \qquad j = 1,....,N \tag{5.24}$$

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} f(m_i) f(m_j) - \sum_{j=1}^{N}\int_0^{m_j} b_j(x) f'(x)dx \tag{5.25}$$

5.3.1 Spiking Activity

Here we consider a BCPNN with spiking activity, *O*, updated asynchronously and stochastically, similar to the Hopfield model, as in eq. (5.26) and (5.27). Here, the differential equation used to update the potential has been dropped. Because we use unary-coded activity, the sum can be moved out from the logarithm in the computation of support (compare Figure 5.2). It is easy to show (Peretto 1984; Kanter 1988; Shim, Kim et al. 1991) that this type of system has stable limit/fix points and that there exists an energy function, eq. (5.28), for it similar to that of the Hopfield network.

$$s_j = \beta_j + \sum_{h=1}^{H}\sum_{i\in Q_h} O_i \log w_{ij} \tag{5.26}$$

$$O_j \leftarrow \begin{cases} 1 & if\ j = \arg\max_{k\in Q_h} s_k \\ 0 & otherwise \end{cases} \qquad h = rand(\{1,...,H\}) \tag{5.27}$$

$$E = -\frac{1}{2}\sum_{h=1}^{H}\sum_{i\in Q_h}\sum_{k=1:k\neq h}^{H}\sum_{j\in Q_k} w_{ij} O_i O_j - \sum_{i=1}^{N}\beta_i O_i \tag{5.28}$$

5.3.2 Continuous-Valued Activity

Next we present the energy function for an approximation of BCPNN with continuous-valued activity, o_i, but without the logarithm of a sum in the support computation. For the exact formulation of BCPNN we have not found an energy function. In this modified network the potential is updated according to eq. (5.29) and the activity is computed, as before, with a softmax function as in eq. (5.30). In appendix IV we prove that this system has fix-point dynamics and an energy function as defined in eq. (5.31).

$$\tau_m \frac{dm_j}{dt} = \left(\beta_j + \sum_{h=1}^{H}\sum_{i\in Q_h} o_i \log w_{ij}\right) - m_j, \qquad j = 1,....,N \tag{5.29}$$

$$o_j = \frac{e^{Gm_j}}{\sum_{k\in Q_h} e^{Gm_k}} : j \in Q_h \ \text{ for each } h = \{1,...,H\} \tag{5.30}$$

$$E = -\frac{G}{2}\sum_{h=1}^{H}\sum_{i\in Q_h}\sum_{k=1:k\neq h}^{H}\sum_{j\in Q_k} w_{ij} o_i o_j - G\sum_{i=1}^{N}\beta_i o_i + \sum_{i=1}^{N} o_i \log o_i \tag{5.31}$$

This system is probably not a special case of the Cohen and Grossberg system since the threshold function, *f* as defined in eq. (5.30), here depends on the potential, m_j, of all units in a hypercolumn. The stability of a fix-point, and how the parameter *G* affects this stability, can be found by linearizing the system around the point. In appendix IV this is done both for BCPNN and the approximate version in eq. (5.29). In particular, using this linearization the critical temperature of the network, i.e. up to which there only exists one attractor in the network, can be estimated. This is useful in optimization applications, because the annealing starts from the critical temperature (Peterson and Söderberg 1989). In eq. (5.30), G is the inverse of temperature.

5.4 Analysis of Storage Capacity

In this section we present analytical signal-to-noise (S/N) analyses of the storage capacity for hypercolumnar networks trained with the Willshaw-Palm and Hopfield learning rules. This type of analysis is used in appendices II, III, and VIII. The networks considered have unary-coded activity in the hypercolumns. The following analyses are restricted to block sparse connection matrices (appendix III), i.e. when connections are deleted, all connections between the units in the pre- and postsynaptic hypercolumns are removed. But it is trivial to perform the analyses also for the case of a randomly diluted connection matrix.

An experimental investigation of the storage capacity in a BCPNN where $H=U=\sqrt{N}$, has shown that the storage capacity in terms of patterns roughly scales as NU (Johansson 2001a). This is in agreement with the theoretical results on Potts neural networks (Kanter 1988; Shim, Kim et al. 1992).

5.4.1 Hopfield Learning-Rule

The working principle of this analysis is to approximate the support for units that participates in the active pattern and also for those that do not, with a normal distribution. This approximation is valid for networks with a large H, because of the central limit theorem. The stability of a network configuration is then computed by calculating the probability that the active unit participating in the active pattern, in each hypercolumn, is set active after one iteration. This analysis only deals with the first step in the iterative retrieval process, but we assume that this first step of retrieval is representative for the qualitative result of iterative retrieval (Schwenker, Sommer et al. 1996).

The analysis starts by calculating the weight-matrix for the Hopfield learning-rule. These weights X_{ij} can be expressed as a sum over Q stochastic variables, one for each training pattern; $X_{ij} = \frac{1}{N}\sum_{\mu}^{Q} X_{ij}^{\mu}$. Using the covariance learning-rule we have three possible outcomes for the contributions to the weight X_{ij} from individual patterns:

$$\begin{aligned}
&1)\quad \Pr(X_{ij}^{\mu} = (1-\tfrac{1}{U})^2) = \tfrac{1}{U^2}\\
&2)\quad \Pr(X_{ij}^{\mu} = \tfrac{1}{U^2}-\tfrac{1}{U}) = \tfrac{2(U-1)}{U^2}\\
&3)\quad \Pr(X_{ij}^{\mu} = \tfrac{1}{U^2}) = \tfrac{(U-1)^2}{U^2}
\end{aligned} \tag{5.32}$$

We will use a normal approximation for the support values for each unit and therefore we calculate the mean and variance of X_{ij}^{μ}, yielding $\mathrm{E}(X_{ij}^{\mu})=0$ and $\mathrm{V}(X_{ij}^{\mu})=(U-1)^2/U^4$. The support can now be stated as a stochastic variable, S_i, that is expressed by a pattern ξ^{μ} and X:

$$S_i = \sum_{k=j:\xi_j^{\mu}=1 \wedge i\neq j}^{K} X_{ik} \tag{5.33}$$

The pattern, ξ^{μ}, is here referred to as the retrieval cue. The support is computed from K terms. The maximum value of K is $H-1$, because the connections within a hypercolumn are always removed. If the connection matrix is sparse, K is smaller than $H-1$. Further, if we want to simulate noise in the retrieval cue, K is reduced. If p_{noise} is the probability of flipping the active unit in a hypercolumn of the retrieval cue, then K should be decreased by Kp_{noise}.

We now treat separately the cases for units participating, denoted by +, and not participating, denoted by −, in the active pattern ξ^μ. In the following, S^+ is the signal support and S^- is the noise support. The mean and variance of S_i^- and S_i^+ are approximated using the central limit theorem, as in eq. (5.34)-(5.37):

$$\mathrm{E}(S_i^-) = 0 \tag{5.34}$$

$$\mathrm{V}(S_i^-) = \frac{QK\mathrm{V}(X_{ij}^\mu)}{N^2} \tag{5.35}$$

$$\mathrm{E}(S_i^+) = \frac{K}{N}\left(1-\frac{1}{U}\right)^2 \tag{5.36}$$

$$\mathrm{V}(S_i^+) = \frac{(Q-1)K\mathrm{V}(X_{ij}^\mu)}{N^2} \tag{5.37}$$

Let F_{S^-} be the normal cumulative distribution function for the support of a unit not participating in the active pattern. Let F_M be the cumulative distribution function of a variable $M=\max_{i\neq j}\{S_j^-\}$ representing the maximum support of those units where i is the unit that participates in the active pattern, then the maximum of the noise supports is:

$$F_M(n)=\Pr(n \geq \max_{j\in Q_h, j\neq i}\{S_j^-\}) = F_{S^-}^{U-1}(n) \tag{5.38}$$

Now we may write the probability that the unit participating in the active pattern has the largest support in a hypercolumn, i.e. the signal support is larger than all noise supports, as;

$$p_{hyper} = \Pr(S^+ > M) = \int_{n=-\infty}^{\infty} f_{S^+}(n)F_M(n)dn \tag{5.39}$$

where f denotes probability density functions and F cumulative probability density functions. The probability that a pattern is stable is $p_{pattern}=(p_{hyper})^H$. The average number of correctly retrieved patterns is $Q_{\mathrm{stored}}=Qp_{pattern}$, where Q is the number of patterns that we have tried to store. We can now determine the storage capacity as the maximum value of Q_{stored} over Q.

Here we have used the distribution functions to calculate the probability of correct recall in a hypercolumn, as in appendix III. This probability can be approximated by using normal distributions with the mean and variances in eq. (5.34)-(5.37) and calculating the probability that a unit participating in the active pattern has a larger support than a unit not participating (appendix II).

5.4.2 Willshaw-Palm Learning-Rule

Here, we perform a similar, one step, S/N analysis of the storage capacity in a network that uses the Willshaw learning-rule. As previously stated, we assume that this is representative for the qualitative result of iterative retrieval. Similar approaches have previously been used (Palm 1980; Buckingham and Willshaw 1993; Schwenker, Sommer et al. 1996; Kartashov, Frolov et al. 1997).

The probability that a certain synapse in the X_{ij} matrix is set to one, when storing a single pattern using the Willshaw learning-rule, is $p_0=1/U^2$. After storing Q patterns, the density of ones in the memory matrix is $p_1=1-(1-p_0)^Q$. We now consider the stability of patterns. Just as in the Hopfield case, we first study one hypercolumn. We calculate the signal support level of the unit that is part of the active pattern, S^+, and the noise support of the other units, S_i^-, as in eq. (5.40) and (5.41). In both cases the support is made up by a sum that run over K weights and at full connectivity we have $K=H-1$. Here, p_{noise} is the probability that the active unit in one hypercolumn of the retrieval cue is flipped. The support levels are expressed as binomial distributions:

$$S^+ \in \text{Bin}(K, 1 - p_{noise}) \tag{5.40}$$

$$S^- \in \text{Bin}(K, p_1) \tag{5.41}$$

Similar to the case of the Hopfield learning rule we can compute the probability for stable recall in one hypercolumn as the sum in (5.42), where F_M is computed from the aggregated binomial cumulative distribution F_{S^-} as in (5.38):

$$p_{hyper} = \Pr(S^+ > M) = \sum\nolimits_{n=1}^{K} f_{S^+}(n) F_M(n-1) \tag{5.42}$$

The probability that all hypercolumns are stable is $p_{pattern}=(p_{hyper})^H$, which is also the expected ratio of stable patterns for a given memory load. As in the previous section we can now determine the storage capacity from $Q_{\text{stored}}=Qp_{pattern}$.

5.4.3 Experimental and Analytical Results

In this section we study how the storage capacity depends on the activity level, noise level in retrieval cues, and connectivity with both experiments and analytical estimates.

In Figure 5.6 experimental and analytically results on the storage capacity are presented for the three different learning-rules; BCPNN, Willshaw, and Hopfield models together with analytical estimates for the Willshaw and Hopfield rules. The networks studied have a constant number of units, N=1000, that are partitioned into different numbers of hypercolumns, H. Changing the number of hypercolumns in a fixed sized network is equal to running a network with different levels of activity. All three rules produce qualitatively similar results, the BCPNN being slightly better than the Willshaw that in turn is slightly better than the Hopfield rule. In Figure 5.6, left, we see that all learning-rules have a peak in the number of stored patterns when the network is partitioned according to $H=\log_2 U$, which was prescribed by Kartashov et al. (1997) as optimal. In the right plot of Figure 5.6 we see that the information capacity per connection has an optimum at $H=\log_2 N$, which is an adaptation of the optimal activity level for non-hypercolumnar associative memories with binary weights (Willshaw, Buneman et al. 1969).

The analytical analysis gives fairly accurate estimates of the storage capacity for the Willshaw rule over the whole range of H values. But in the case of the Hopfield rule, the analytical analysis shows poor agreement with experimental results for small H values. The reason being that the assumption of the central limit theorem, which was used in the derivation of the analytical estimate for the Hopfield rule, is not met. The central limit theorem requires that the synaptic sum be composed of a large number of terms, which is not the case if H is small. Further, this means that if the connectivity is low, or if there is noise in the retrieval cues, the analytical estimates will deviate from the correct values of the storage capacity for the Hopfield network.

In Figure 5.7 the storage capacity is plotted for the same networks as used in Figure 5.6, but now noisy retrieval cues are used. The optimal number of H is increased, which means that the optimal activity level is also increased. The increased activity level adds redundancy to the stored patterns, which is necessary when the retrieval cues contain noise. Another aspect of using noisy retrieval cues is that the storage capacities for both the Willshaw and Hopfield rules are underestimated by the analytical analyses. The reason being that the analyses only approximates the first step of the networks' retrieval process and not the complete relaxation process.

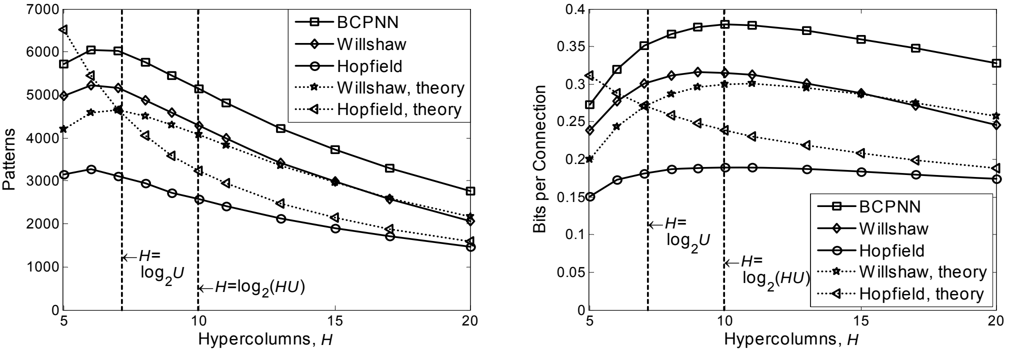


Figure 5.6. Storage capacity plotted as a function of *H*, for networks with *N*=1000. The x-axis can be interpreted as the activity level; *a*=*H*/*N*. Noise free retrieval cues are used.

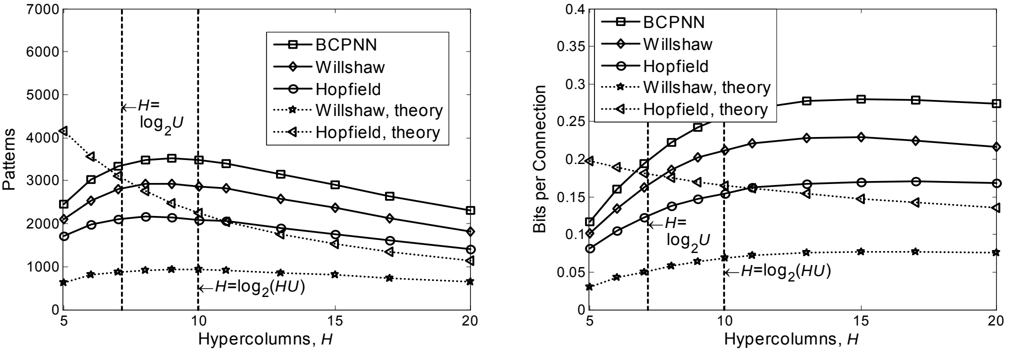


Figure 5.7. See Figure 5.6. Here, retrieval cues with 20% noise are used.

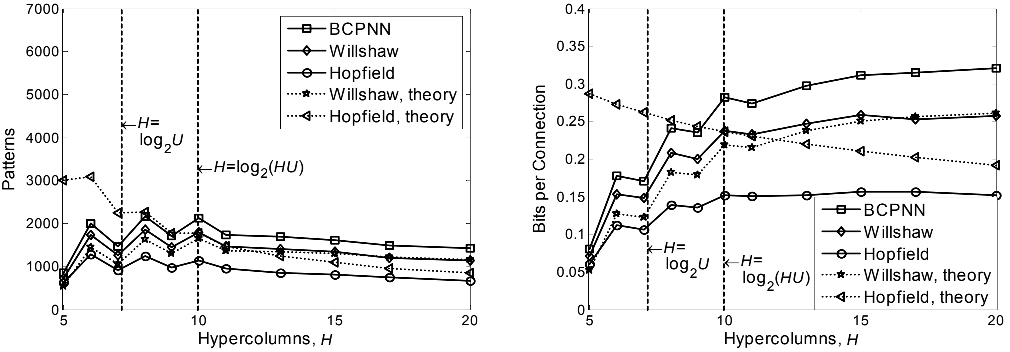


Figure 5.8. See Figure 5.6. Here, networks with *C*=50% connectivity and noise free retrieval cues are used. The jumps in the curves are due to truncation effects from approximating 50% connectivity with a small number of hypercolumns.

Figure 5.8 shows the storage capacity plotted as a function of the connectivity. Similar to the case of noisy retrieval cues, we see that a decreased connectivity gives a shift of the optimal storage capacity towards large activity levels, i.e. many but small hypercolumns. A heuristic rule for partitioning the network, and setting the activity level, in the case of decreased connectivity is $H=(\log_2 N)/C$, where C is the connectivity. From the literature we know that a decrease in connectivity incurs a decrease of the storage capacity because of a decreased signal-to-noise ratio caused by the missing connections (Maravall 1999).

5.5 Cortex as an Attractor Network

Over the last couple of decades it has been speculated that global states of activity can persist in the neocortex which we in this thesis to a first, very rough, approximation model by a modular attractor network (Little and Shaw 1975; Palm 1980; Palm 1981b; Fransén and Lansner 1998; Rolls and Treves 1998; Lansner, Fransén et al. 2003). Serious critic of this idea has been presented by O’Kane and Treves who found that the number of unique and stable states, i.e. memories, in such an attractor network does not scale with the number of units but with the number of connections per unit (O'Kane and Treves 1992a; O'Kane and Treves 1992b). From small to large cortices the number of connections per neuron is roughly constant and the number of memories that can be stored is therefore limited to the order of thousands (Treves 2005). More specifically, O’Kane and Treves studied multi-modular networks where subgroups, i.e. modules, of neurons instantiate small attractor networks, which in turn are connected by long-range connections to form a global attractor network that spans over the entire cortex. In a subsequent study by Fulvi Mari and Treves two modifications of these global attractor networks were proposed that greatly increased the number of memories that can be stored and which made their count scale with the number of units and connections (Fulvi Mari and Treves 1998). These two modifications are; firstly, the patterns of activity (memories) are no longer random but are matched to the connectivity or vice versa, and secondly, simultaneous activity is only allowed in a fraction of all modules and the remaining being put into a quiescent state, i.e. a memory pattern sparsely activates the modules.

In order to enable simulation studies of the networks discussed by Fulvi Mari and Treves, Kropff and Treves have proposed an abstract version based on a Potts-network (Kropff and Treves 2005; Treves 2005). In this abstract version of O’Kane and Treves model, a Potts unit is used to represent each module and the different states of the Potts-unit represent certain attractor states in the module. Our cortical model is instantiated with a network similar to that of Kropff and Treves but we map it differently and more explicitly onto the neocortex (appendix VIII). In our network each Potts-unit is represented by a hypercolumn and each of the Potts-unit’s different states corresponds to a minicolumn.

In our abstract cortical model each functional unit, i.e. minicolumn, has on the order of 10^5 incoming connections. Thus, a network of minicolumns has a higher connectivity than a network of neurons. In the latter case each unit only has on the order of 10^3 incoming connections. The higher connectivity increases the number of memories that can be stored but it does not completely solve the scaling problem pointed out by O’Kane and Treves. But by allowing hypercolumns to be quiescent the scaling problem is solved and we get a storage capacity for cortical sized networks that is on the order of millions (appendix VIII).

5.6 Patchy Connectivity

A characteristic feature of cortical connectivity is its patchiness and in appendix III we have used the abstract cortical model to investigate this aspect of cortical connectivity. In the cortical model patchy connectivity is defined as a connectivity scheme in which all units or none connects to either a pre- or postsynaptic unit. In appendix III three different types of patchy connectivity are investigated: *Unit-to-hyper* patchiness that applies to the outgoing connections; one minicolumn projects either to all minicolumns in a hypercolumn or to none. *Hyper-to-unit* applies to incoming connections; one minicolumn tends to receive connections from either all units in a hypercolumn or from none. *Block* which is a combination of the first two making connections between hypercolumns reciprocal, i.e. all units in the pre- and postsynaptic hypercolumns are connected. These different types of connectivity were investigated both experimentally and with signal-to-noise analyses for both the Hopfield and Willshaw learning-rules. We found that for the hyper-to-unit and block type of connectivity the storage capacity was increased compared to the case of random connectivity. This was explained by the analysis with an improved signal-to-noise ratio for patchy connectivity. Further, experiments on non-hypercolumnar networks showed that they also got an increased storage capacity with patchy connectivity.

This is interesting because as previously stated the connectivity in cortex has a patchy layout, which has been seen in experiments with both retro- and anterograde tracers (Boyd and Matsubara 1991; Conway, Boyd et al. 2000; Galuske, Schlote et al. 2000). In addition to the computational advantage, which we show that patchy connectivity gives, it also reduces the total wiring length if multiplexed transmission lines are used (Mitchison 1992). This both reduces volume and saves energy.

We have also experimented with the BCPNN and got qualitative similar results to those for the Willshaw and Hopfield rules, i.e. that patchy connectivity improves the storage capacity.

This study is a good example of how an abstract cortical model can be used to investigate and explain the anatomy of cortex.

5.7 Small-World Connectivity

In this section we discuss networks with small-world connectivity (Watts and Strogatz 1998; Barabási and Albert 1999; Newman 2003), who's characteristics first were investigated in the end of the 1990's and presented in a seminal paper by Watts and Strogatz (1998). Since then, this type of connectivity has been found to exist in a wide range of different types of networks, everything from power grids to networks of nerve cells. Networks with small-world type of connectivity are characterized by a small average path-length together with a high degree of clustering. A small average path-length means that in a network of nodes it is possible to go between two arbitrary nodes only passing a small number of nodes on the way. A high clustering means that if a node A is connected to both node B and C there is a high probability for a connection also between nodes B and C. Here, we first make a short review of work done on attractor networks with small-world connectivity, then we present results from our own studies of small world connectivity in the abstract cortical model.

Hopfield networks that have a small world connectivity and are used with densely coded patterns have been studied by a number of authors (Bohland and Minai 2001; McGraw and Menzinger 2003; Morelli, Abramson et al. 2004). Further, the stability of a Hopfield network formulated as dynamical system has been studied (Li and Chen 2003), and also networks with integrate and fire neurons have been studied (Masuda and Aihara 2004). But attractor networks with sparsely coded patterns and small-world connectivity have not been studied elsewhere to our knowledge (Johansson and Rehn 2004). There are a number of motifs for studying attractor networks with this type of connectivity; it has been suggested that it better matches the connectivity of nerve cell networks than either local or random connectivity. It has also been argued that it enables saving of considerable amounts of wiring (Bohland and Minai 2001; Morelli, Abramson et al. 2004). Further, small-world connectivity has been studied in the context of attractor networks' storage capacity (Bohland and Minai 2001; McGraw and Menzinger 2003; Kim 2004; Morelli, Abramson et al. 2004).

In appendix III we investigate if the networks we use with patchy connectivity (section 5.6) have small-world characteristics. In the case of 100% patchiness, these networks have a structured connectivity that gives them a 50% higher degree of clustering than random ones. Also, because the patches of connections are randomly distributed, the average path-length is almost the same as for random connectivity. Taken together, this puts the networks with patchy connectivity in the small-world domain. The main result in appendix III is that patchy connectivity is better than random because it gives a better S/N ratio, which in turn results in a higher storage capacity.

Next we study five differently sized networks instantiating the abstract cortical model, but without patchy connectivity (Johansson and Rehn 2004). Here we assume that 5 synapses are used per connection and that a minicolumn has a total of $1.2 \cdot 10^5$ connections. In each minicolumn there are $4 \cdot 10^4$ local connections that connect to nearby minicolumns and $8 \cdot 10^4$ global connections that connect to arbitrary minicolumns in the network. In Table 5.1 we have computed the average path-length and clustering as defined by Watts and Strogatz (1998), and we found that the minicolumn-networks only deviated slightly from randomly connected networks in respect to the average path-length. Experiments with extremely local networks showed that even when only 100 of the $1.2 \cdot 10^5$ connections per unit were global, the average path-length in the cat-network was no more than 2.04. The relative difference in clustering between the actual and random connectivity was largest in the large networks. In summary this means that the small-world property is a feature only present in the larger networks corresponding to human, macaque, and cat, given this type of non-patchy connectivity.

Table 5.1. Average path-length and clustering computed for the abstract cortical model without patchy connectivity and equally sized networks with random connectivity.

	Human	**Macaque**	**Cat**	**Rat**	**Mouse**
Minicolumns	$2.0 \cdot 10^8$	$2.0 \cdot 10^7$	$6.1 \cdot 10^6$	$5.0 \cdot 10^5$	$1.6 \cdot 10^5$
Average Path-Length Small-World	2.00	2.00	1.98	1.81	1.54
Average Path-Length Random	2.00	2.00	1.98	1.78	1.55
Clustering, Small-World	0.047	0.059	0.072	0.25	0.56
Clustering, Random	0.00060	0.0060	0.020	0.21	0.53
Relative Clustering, Small-World / Random	79	9.8	3.7	1.2	1.1

5.8 Diluted Connectivity

A typical feature of cortical neural networks is that they are not fully connected; each neuron has a fairly fixed number of connections. This is due to physical constraints such as packing volume and metabolism. Also in an engineered system there is an obvious advantage in only representing a selected number of important connections.

In a master thesis work, Sandh has investigated the characteristics of sparsely connected BCPNN:s and found that they have a slightly reduced storage capacity per connection compared with fully connected networks and he has also confirmed that a slightly asymmetric weight-matrix does not reduce the storage capacity significantly compared with a perfectly symmetrical one (Sandh 2004). These findings are in agreement with what is known about the archetypical Hopfield network (section 4.7).

Further, Sandh has investigated possible ways in which a weight-matrix can be pruned in order to achieve a higher storage capacity than that of a randomly diluted network. Sandh identified two different ways in which this could be done: Either the negative weights are removed first or the weights that are closest to the mean of all weights are removed first. Sandh concluded that the latter method is better and that it gives the highest storage capacity.

So far in this section we have discussed networks that potentially can have a connection between any pair of units and during training these networks have full connectivity. In an implementation of a sparsely connected network we never want to store more than a small fraction of all possible weights. Sandh has experimented with BCPNN:s that are only allowed to have a limited number of weights per unit. In order for this type of networks to be able to learn an arbitrary pattern the connections have to be dynamically relocated during training. He developed an algorithm in which a fixed number of weights are continuously reallocated during training. In this algorithm the weights are at first randomly distributed in the network. After training a pattern, weights close to the mean of all weight-values (i.e. zero) are randomly reallocated. The drawback with this approach is that it requires the set of training examples to be presented repeatedly and that the algorithm sort the weights according to their strength. In a current master thesis work by Alexander Lindquister a more refined method is developed.

5.9 Memory Clustering

In appendix IV we characterize the clustering of stored patterns in a BCPNN. We do this by experiments and analytically by linearizing the system around fix-points i.e. stored patterns. In particular we study the effects resulting from changes in the gain parameter, G, in the network's softmax function. Clustering of memories has previously been studied by Eriksson and Lansner (Eriksson and Lansner 2003) and by Levin (Levin 1995). The most important finding is that the number of attractors (memories) in the network can be controlled by the G parameter. The maximal storage capacity is achieved for high values of G, and when G is decreased the number of fix-points decreases as well. For sufficiently low values of G, only one attractor is present.

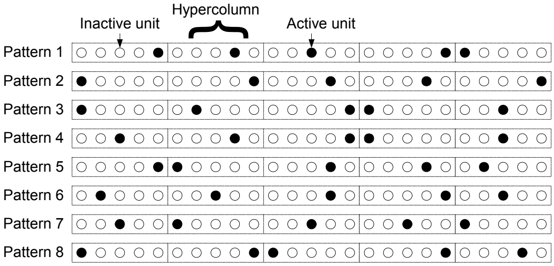
Clustering of the memories is a very powerful operation that can be useful in several applications. As shown in appendix IV, the number of clusters in an attractor network can be altered very rapidly without having to go through the training data a second time or update any of the parameters set during learning (weights and biases). The process of learning is generally also much faster in an attractor network than in other self-organizing clustering algorithms (Orre, Bate et al. 2005). The Achilles heal is that attractor clustering do not obey an explicit metric as in traditional clustering algorithms. In an attractor network the clusters are formed based on an intricate and implicit relationship between the stored patterns and the network's dynamics.

5.10 Attractor Size Modulation

Here, we discuss two ways in which the size of attractors in a BCPNN can be modulated (Johansson and Lansner 2005); firstly, by training one or several patterns repeatedly and, secondly, by adjusting the noise level in one or several patterns. In the latter case several copies of each pattern is trained. In appendix IV we show that the attractor size can also be modulated by the *G* parameter in the softmax function.

5.10.1 Repeated Training of Patterns

In these experiments we studied how repeated training of a few selected patterns modified the attractor size. We used a BCPNN with H=5, U=5, τ_c=10, and G=2, that was trained with eight randomly chosen prototype patterns shown in Figure 5.9. The attractors found in the network were often slightly shifted from the unary-coded prototypes, because of the relatively low *G*-values. The attractors were classified as corresponding to one of the prototypes by measuring the L_2 norm (Euclidean distance) and selecting the prototype to which the distance was the smallest. The size of the attractors was measured by counting the number of retrieval cues of all possible permutations (5^5=3250 cues) that converged to the attractor. No spurious attractors were found for this particular set of patterns, which also was used in the experiments in appendix IV.



Pattern 1	○ ○ ○ ○ ●	○ ○ ○ ● ○	○ ○ ● ○ ○	○ ○ ○ ○ ●	● ○ ○ ○ ○
Pattern 2	● ○ ○ ○ ○	○ ○ ○ ○ ●	○ ○ ○ ● ○	○ ○ ○ ● ○	○ ○ ○ ○ ●
Pattern 3	● ○ ○ ○ ○	○ ● ○ ○ ○	○ ○ ○ ○ ●	● ○ ○ ○ ○	○ ○ ● ○ ○
Pattern 4	○ ○ ● ○ ○	○ ○ ○ ● ○	○ ○ ○ ○ ●	● ○ ○ ○ ○	○ ○ ● ○ ○
Pattern 5	○ ○ ○ ○ ●	● ○ ○ ○ ○	○ ○ ○ ● ○	○ ○ ○ ● ○	○ ● ○ ○ ○
Pattern 6	○ ● ○ ○ ○	○ ○ ● ○ ○	○ ○ ○ ● ○	○ ○ ○ ○ ●	○ ○ ● ○ ○
Pattern 7	○ ○ ● ○ ○	● ○ ○ ○ ○	○ ○ ● ○ ○	○ ○ ● ○ ○	● ○ ○ ○ ○
Pattern 8	● ○ ○ ○ ○	○ ○ ○ ○ ●	● ○ ○ ○ ○	○ ○ ○ ○ ●	○ ○ ○ ● ○

Figure 5.9. The eight prototype patterns that the BCPNN was trained on.

Figure 5.10 shows how the sizes of all eight attractors changed when pattern 8 was trained repeatedly. The attractor corresponding to pattern 3 disappeared when the number of repetitions of pattern 8 was larger than four. Probably this attractor was combined with that of pattern 4, because these two patterns are very close in terms of hamming distance. The size of attractor 8 increased a little following the first repetition, after which it then decreased. Here, we have used the BCPNN learning-rule based on counting, but if the learning-rule based on exponentially weighted moving averages had been used the increase in attractor size would have been more pronounced (Sandberg and Lansner 2001). Training a prototype pattern repeatedly is equivalent to regulate the parameter κ during learning in eq. (5.7).

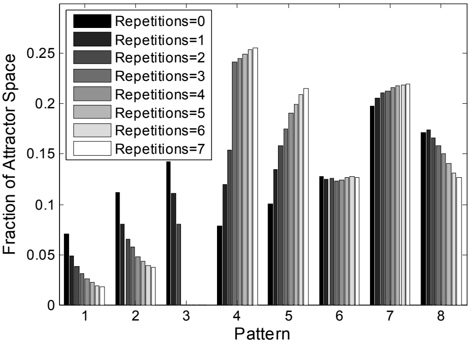


Figure 5.10. The changes in size of all attractors when pattern 8 is repeatedly trained.

5.10.2 Noisy Training Patterns

The size of an attractor depends on the noise level in the presented prototype patterns. Here we investigate the size of the attractors in a network that is trained with five different prototype patterns that each is presented to the network twenty times. One of these prototypes has noise added, meaning that it show up slightly different in each of the twenty presentations (Figure 5.11, left). In Figure 5.11, right, the sizes of the attractors corresponding to the five prototypes are plotted for varying levels of noise. A small level of noise increases the attractor size, but a high level of noise reduces the attractor size. The maximum size of attractor 5 is attained at 50% noise. Pattern 6 in the figure represents all spurious fix-points in the network that were not associated with any of the attractors corresponding to stored prototypes.

The network used in this experiment had H=5, U=5, τ_c=10, and G=2. Figure 5.11, left, shows an example of the 100 patterns constituting the training set. Here, 50% noise was added to the twenty instances of prototype 5.

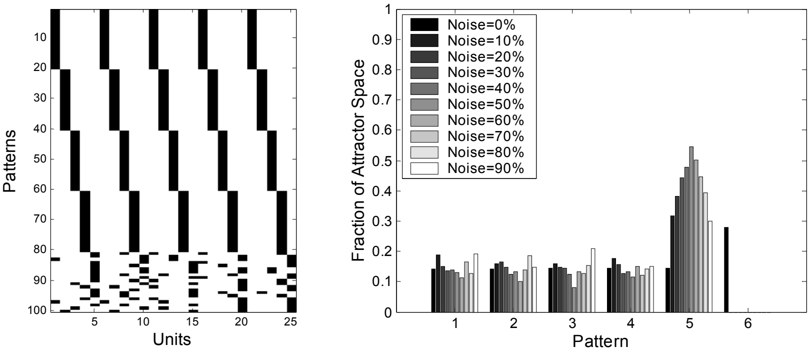


Figure 5.11. Left: The training set consisting of twenty instances of each of the five different patterns. The particular training set shown has 40% noise in the fifth prototype. Right: The size of each of the five attractors plotted for different levels of noise in the fifth prototype.

6 Towards a Brain Inspired Computing System

In this chapter we describe a multi-layer implementation of the hierarchical cortical model (section 3.4) and discuss its computational capabilities. We show that this type of multi-layer networks can be used as both content addressable memories and classifiers with favorable results. Important and characteristic features of systems based on this architecture are massive parallelism, redundancy, noise resistance, and plasticity in terms of both learning and self-organization.

Here we use image data as the results then are easy to visualize, but the computations performed by the network are of a general type and apply equally well to other types of data. For more information on neural networks applied to image processing we refer to an excellent review by Egmont-Petersen et al. (Egmont-Petersen, Ridder et al. 2002).

In this chapter we mainly present the results from appendices VI and VII, but we also discuss one of the results from appendix I.

6.1 Autoassociative Memory with an Attractor Network

Much of the work in this thesis is focused on autoassociative memory implemented with attractor neural networks. A memory of this type can perform impressively on tasks such as noise reduction, pattern completion, and prototype extraction given that the data is encoded in a suitable way. This usually means that the encoded patterns should be sparse, i.e. only a small fraction of all units are active, and neither should the patterns have high correlations. In this section we first present results from experiments where patterns that meet the sparseness requirement are used, and then we exemplify by experiments the situation where this condition is not satisfied. For the former case we first describe experiments testing noise reduction and pattern completion and then we present an experiment testing prototype extraction. We used a fully connected BCPNN with spiking units that had the parameters; H=1024, U=16, τ_P=3000, τ_M=10 (see section 5.2).

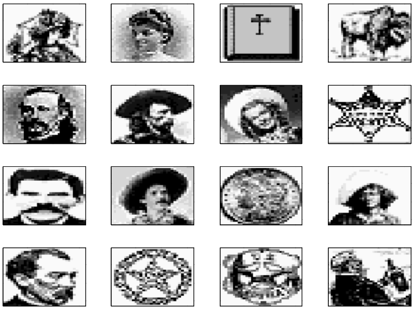


Figure 6.1. 32×32 pixel images with 16 levels of gray.

In the first experiment, testing noise reduction and pattern completion, we trained the network on the images shown in Figure 6.1. Each pixel was represented by a hypercolumn with 16 units, one for each gray-level, and this gave an activity level of a=1/16. Obviously, these patterns (images) are not random and are likely to be slightly more correlated than such. Although, the network manages to fully recover the stored images from the retrieval cues with salt and pepper noise (Figure 6.2, left) and also from partially occluded views of the images (Figure 6.2, right).

In the second experiment, we trained the network with a hundred different images, all with 50% salt and pepper noise. To test the network, we used the noise free images in Figure 6.1 as retrieval cues. The result was that all of these images were stored as stable fix-points, although the network had never seen any of them without noise. This is an example of an attractor network doing prototype extraction, i.e. creating a holistic view from fragmented data. For more information on attractor networks performing this type of function, see Orre et al. (Orre, Bate et al. 2005).

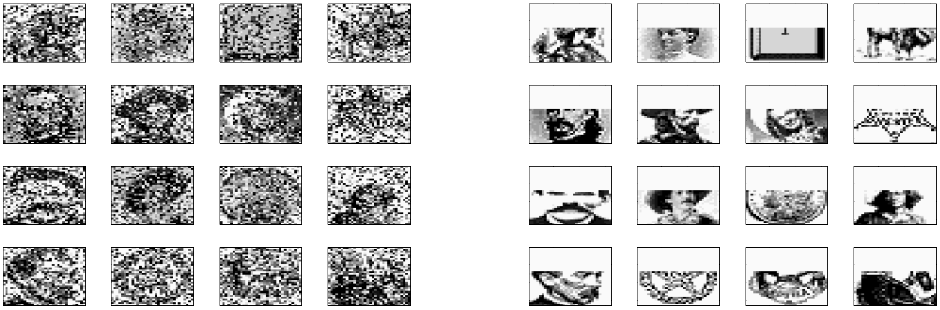


Figure 6.2. Images used as retrieval cues. Left, the images in Figure 6.1, but with 50% salt and pepper noise. Right, the same images but with 40% of their area occluded.

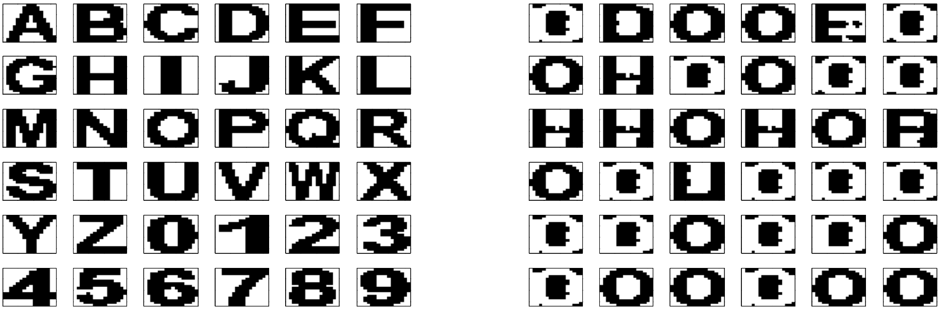


Figure 6.3. Left, 16×16 pixel black and white images of letters and numerics taken from the font Arial. Right, the output from a BCPNN trained on the images to the left, after presenting the network with the same noise free images.

Next we give an example taken from appendices VI and VII of a situation in which a single layered attractor network is less successful. In this experiment we again used a fully connected BCPNN that had the parameters; H=256, U=2, τ_M=10. This network was trained using the counting version of the learning-rule. In this experiment we stored the black and white images of letters shown in Figure 6.3, left. Each pixel was represented by a hypercolumn with two units, one representing black and the other white as shown in Figure 6.4. This gave an activity level of a=1/2 in the network. On retrieval the stored, noise free, images were used as retrieval cues. For each input the network was iterated to a stable state. These states, i.e. the outputs, are shown in Figure 6.3, right. From this figure we can conclude that the network was not able to store any of the trained images.

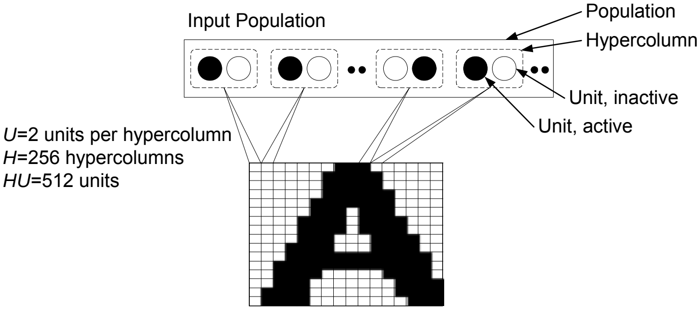


Figure 6.4. Each pixel in an image is coded by a hypercolumn in the network.

6.2 A Computational Instance of the Hierarchical Cortex Model

We start by presenting the conceptual idea behind the hierarchical model presented in section 3.4 and next we present an implementation of a subsection of it. There are two processing streams in the system: The lower levels extract features that are used to represent the input data. These features are conveyed to the higher levels in the hierarchy. The higher levels combines inputs from several different lower levels in the system and form a holistic interpretation. The higher levels then feed the lower levels with top-down predictions. This system is somewhat similar to other hierarchical processing models of cortex (Dayan, Hinton et al. 1995; Rao and Ballard 1997; Deco 2001; Lee and Mumford 2003; Deco and Rolls 2004; Hawkins 2004; Dean 2005; George and Hawkins 2005).

According to Linsker's Infomax principle the feature extraction algorithm should try to maintain as much information as possible about the input data. A powerful and commonly used approach to feature extraction, which adhere to the goals of information preservation, is to use multiple layers of hierarchically arranged feed-forward networks that implements competitive learning (Linsker 1986b; Linsker 1986a; Linsker 1986c; Grossberg 1987; Fukushima 1988; Linsker 1988; Fukushima 1989; Fukushima and Wake 1991; Földiák 1991; Togawa, Ueda et al. 1991; Wallis and Rolls 1997; Rolls and Treves 1998). This type of structures can also achieve invariant pattern recognition, e.g. with slow learning (Földiák 1991). Based on arguments of neural energy efficiency it can be argued that the code should be sparse (Olshausen and Field 1997; Olshausen and Field 2004). The advantages with sparse coding have been demonstrated in several papers (Michaels 1996; Peper and Shirazi 1996; Bartlett and Sejnowski 1998; Amit and Mascaro 2001).

The top-down projection has been suggested to have a number of possible roles: Firstly, it can be used to represent a prior or a model of the input (Lee and Mumford 2003). Such a prior can be used to actively filter the input based on expectations, a good example being the cocktail party problem where the task is to discriminate what one speaker out of many says. In this situation, the signal-to-noise ratio is very low. Here the top-down connection could provide a model for a particular person's speech, which would improve the signal-to-noise ratio significantly. Secondly, top-down information could be used to steer attention, i.e. implement biased competition (Deco and Lee 2002). Thirdly, these connections could be used to substantially limit the number of object representations that need to be considered and thereby speed-up the processing (Bar 2003). Finally, these projections could also take part in, and improve, memory recall (Renart, Parga et al. 1999).

6.2.1 Implementation of a Section Taken From the Hierarchical Model

In this section we present a computational implementation of a part of the hierarchical cortical model as shown in Figure 6.5 and in particular this instance is based on the interactions between two populations. It is straightforward to generalize the proposed implementation to a hierarchical setup, although it is not done here. The input population has a forward projection, projecting up into the hypercolumns of the hidden population. This projection is trained with competitive learning, e.g. competitive selective learning (CSL) (Ueda and Nakano 1994). CSL can theoretically achieve a partitioning that is optimal in the sense that it minimizes the variance of the input data representation. The hidden population has a recurrent projection and a feedback projection down onto the input population. These projections are trained with a Hebbian learning-rule, e.g. BCPNN.

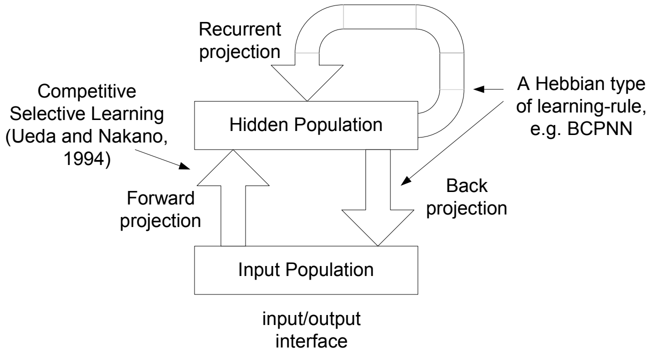


Figure 6.5. A computational instance with two populations taken from the hierarchical cortex model. A population corresponds to a patch of cortex with a size up to that of a cortical area.

The forward projection, that implements competitive learning, is composed of several projections that go from groups of hypercolumns in the input population to hypercolumns in the hidden population (Figure 6.6). These groups of hypercolumns in the input population are here referred to as a *receptive fields*. The units in the hypercolumns in the receptive field form an input vector $\boldsymbol{x}^h$ that is used in the competitive learning. Figure 6.6 shows an input population with three receptive fields. The receptive field h=2 in center is composed of eight hypercolumns with two units in each and the corresponding input vector $\boldsymbol{x}^h$ is composed of 16 elements. Every one of these units connect to each and one of the units in the hidden hypercolumn indexed with h=2 and for each unit i in this hidden population a weight vector $\boldsymbol{w}_i^h$ is updated by eq. (6.1). This is how the model is implemented in appendix VII.

$$\boldsymbol{w}_i^h = \begin{cases} \boldsymbol{w}_i^h + \dfrac{\boldsymbol{x}^h - \boldsymbol{w}_i^h}{\tau_C} & \text{if unit } i \text{ wins the competition in hidden hypercolumn } h \\ \boldsymbol{w}_i^h & \text{otherwise} \end{cases} \tag{6.1}$$

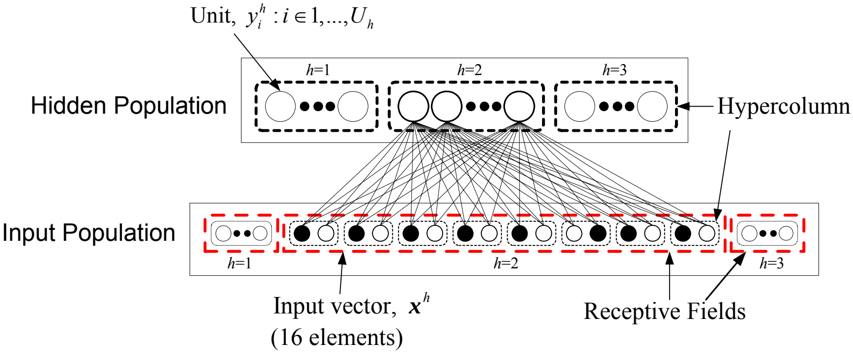


Figure 6.6. Schematic view of how hypercolumns in the input population are grouped into receptive fields and how the units in a receptive field project into a hypercolumn in the hidden population.

For certain types of input data that is unary coded, i.e. a single winner in each hypercolumn, the competitive learning can be made more efficient by reducing the number of dimensions in the input vector $\boldsymbol{x}^h$. The idea is to replace the activity of each unit with a scalar value that codes for the active unit in each hypercolumn. In the example shown in Figure 6.7 the number of elements in the input vector $\boldsymbol{x}^2$ is reduced from 16 to 8 using this optimization. Each element in the input vector $\boldsymbol{x}^h$ is coded onto an interval (0,1) by $i/(U-1)$ where i is the index of the most active unit in a hypercolumn. The number of elements in $\boldsymbol{x}^h$ now equals the number of hypercolumns in the receptive field. In appendices I and VI the model is implemented in this way.

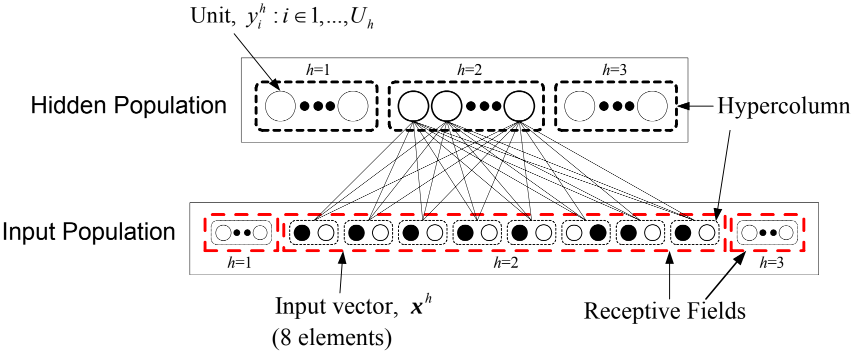


Figure 6.7. An optimized version of the system in Figure 6.6 where a more compact coding is used in the input population. Here, the distribution of activity in each hypercolumn is replaced with a scalar.

From an algorithmic perspective, implementing competitive learning in many small sub-regions of the input, i.e. the receptive fields, instead of applying it to the entire input is a divide-and-conquer strategy. A big advantage with using this strategy is that its fast and its trivial to make an efficient parallel implementation of it.

The system that we discuss here can be seen as an attractor network with a preprocessing stage that decorrelates data. Such a system has similarities to one studied by Bartlett and Sejnowski for invariant face recognition (Bartlett and Sejnowski 1998). Similar to our system, competitive learning is used to create a projection from an input to a hidden population. Also, as in our system, the hidden population implements an autoassociative memory, but there is no back-projection from the hidden to the input population in Bartlett and Sejnowski's network. The idea of adding a feature extracting preprocessor to an associative memory has also been explored by other authors (Michaels 1996; Peper and Shirazi 1996). There are also authors that have simply skipped the feature extraction step and used a system with a random sparse projection from the input to the hidden population in order to decorrelate data (Amit and Mascaro 2001).

6.2.2 Formation of Receptive Fields using Agglomerative Clustering

Deciding which hypercolumns to group together in the input population, i.e. forming the receptive fields, can be done in a number of ways. A few alternative examples are; they can be set up randomly, they can be organized according to the statistics of the input data, or they can be set up according to some heuristic scheme. In case the purpose of the forward projection is to efficiently recode the input data into a non-redundant code, such as in Linsker's Infomax principle (Linsker 1988), it is important that the receptive fields are formed by hypercolumns that process correlated and similar information. In the case of image data, setting up receptive fields organized as square regions use the heuristic knowledge that nearby pixels in an image are correlated. But for many other types of data, e.g. data from process industries, the dependency structure is unknown and must be statistically inferred. Here we use an algorithm (Kraskov, Stögbauer et al. 2005) that cluster the input data based on a mutual information based distance measure D, eq. (6.2), where X and Y are probability density functions each representing a cluster.

$$D(X,Y) = 1 - \frac{I(X,Y)}{H(X,Y)} \tag{6.2}$$

The mutual information, I, and entropy, H, between two hypercolumns (clusters) X and Y can easily be computed if the BCPNN weight-matrix, with the p_{ij} and w_{ij} variables, has been computed:

$$I(X,Y) = \sum_{i \in Q_X} \sum_{j \in Q_Y} p_{ij} \log_2 w_{ij} \tag{6.3}$$

$$H(X,Y) = -\sum_{i \in Q_X} \sum_{j \in Q_Y} p_{ij} \log_2 p_{ij} \tag{6.4}$$

The clustering algorithm by Kraskov et al. is agglomerative, meaning that it starts out with all hypercolumns as being individual clusters, i.e. each receptive field only contains one hypercolumn. The algorithm then successively joins clusters according to the distance measure in eq. (6.2), thus reducing the number of clusters (and hypercolumns). For each new cluster formed, the activity must be normalized and the distances to all other clusters must be computed.

Because all hypercolumns in the hidden population have an equal number of units it is important that all clusters contain an equal amount of information. In the original clustering algorithm, one cluster can contain much more information than all other clusters, which results in a high reconstruction error for the part of the input covered by that particular cluster. In the following experiment we have used a modified version of Kraskov's algorithm. In this version we have added a heuristic in order to make all clusters have similar entropies. In the modified algorithm, the cluster X with the smallest entropy, $H(X,X)$, is always joined with another cluster (according to the distance measure).

Kraskov's clustering algorithm does not generate overlapping receptive fields, i.e. hypercolumns in the input population can only participate in one cluster. We have experimented with overlapping receptive fields (appendix VI). It is likely that the receptive fields in the visual system have overlaps and that these overlaps are important for creating e.g. size invariance.

6.2.3 Neocortex as a Hierarchical Bayesian Inference Machine

The proposed cortical model (section 3.4) when used in a hierarchical setup could be regarded as an instantiation of the hierarchical Bayesian inference machinery proposed by Lee and Mumford as a model of the visual system (Lee and Mumford 2003). In their framework the computations do not proceed in a bottom-up serial fashion but occur concurrently in a number of loops throughout the entire hierarchy (Figure 6.8). The early stages in this processing do more than simple feature extraction. Rather, they participate continuously in the computations and alter their state in response to changes in the prior believes. Lee and Mumford's framework implements something similar to non-parametric loopy belief propagation (Dean 2005), which is possible to implement with neural circuits (Shon and Rao 2005).

Lee and Mumford do not propose any implementation, but they suggest that population coding of the probability distributions could be used in an eventual implementation. George and Hawkins propose a system that in many aspects is an instance of this framework (George and Hawkins 2005). But their model lacks a biologically plausible learning procedure; they use an algorithm that memorizes all possible instances of the training data. Our hierarchical model, on the other hand, only implements biological realistic learning procedures and it also has a non-parametric representation of probability distributions. It is therefore an excellent candidate for implementing Lee and Mumford's framework.

There are a number of neurophysiological experiments that show that the feed-back connections in the visual system facilitate the neural response in lower levels when the inputs to these primary levels are in accordance with the global percept (Hupe, James et al. 1998; Lee and Nguyen 2001). There are also models such as the adaptive resonance theory that works in this way, increasing the strength of the active units when the input matches the system's state or prior expectations (Carpenter and Grossberg 1987).

However, there are experiments that show a decrease in neural activity in the primary stages of the cortical visual processing hierarchy when the input matches prior believes (Murray, Kersten et al. 2002; Murray, Schrater et al. 2004). This effect is explained such that feedback from higher areas serves to suppress activities in earlier areas, only allowing the error between the prior and the input to be propagated to the next level. This idea is referred to as *explaining away* (Rao and Ballard 1997; Rao and Ballard 1999).

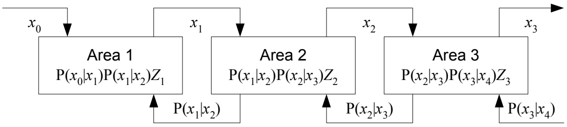


Figure 6.8. A schematic of the hierarchical Bayesian framework proposed by Lee and Mumford (2003). The regions of cortex are linked together in a Markov chain. Z is a normalization factor. In case of the visual cortex areas 1, 2, and 3 could correspond to visual areas V1, V2, and V4 respectively (Lee and Mumford 2003) and in case of our model population 1, 2, and 3 respectively.

6.3 Autoassociative Memory Based on the Hierarchical Cortex Model

Here we present some results from the experiments in appendix VI and VII, where systems based on the architecture presented in section 6.2 implement autoassociative memory.

6.3.1 Results on Letters

In section 6.1 we concluded that a single layer attractor neural network is incapable of processing and storing correlated patterns (the 36 black and white images of different letters). To overcome this deficit and to build an autoassociative memory that also can handle this type of patterns we have implemented an instance of the system shown in Figure 6.5. The images are encoded in the input population as shown in Figure 6.4. This type of system has the typical abilities of a regular attractor neural network such as noise reduction, pattern completion, and prototype extraction but most importantly it can store correlated patterns such as those shown in Figure 6.3, left. The purpose of the mapping from the input to the hidden population in this system is to reduce correlations (and redundancies) and sparsify the input data, which is achieved by means of competitive learning.

We have experimented with different partitioning schemes of the receptive fields (Figure 6.9), one data dependent (*informed*) and three data independent (*heuristic*, *random fan-in*, *random fan-out*). The informed partitioning scheme uses the mutual information based clustering described in section 6.2.2 to generate receptive fields. The heuristic partitioning sets up the receptive fields such that spatially nearby pixels, which are strongly correlated, are grouped together. In appendix VI we experimented with overlapping receptive fields and the heuristic partitioning scheme was then implemented with line segments instead of squares for reasons of simplicity. Random fan-in sets up the receptive fields such that all units in the hidden population connect to the same number of hypercolumns in the input population. Random fan-out sets up the receptive fields so that all hypercolumns in the input population connect to the same number of units in the hidden population. On all instances where the number of pixels per receptive field times the number of receptive fields divided by the total number of pixels is an integer, the two random partitioning schemes are identical.

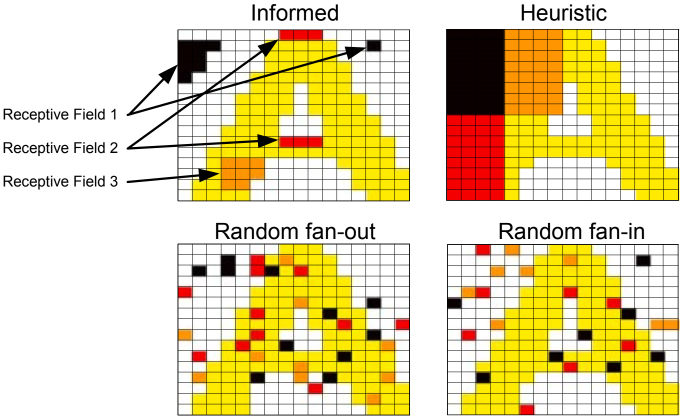


Figure 6.9. Four different partitioning schemes of the receptive fields. Three different receptive fields are shown with different color codes. The receptive fields created by the heuristic partitioning scheme contains more pixels than the other three.

6.3.2 Results on Pictures

Here we present results from appendix I on using the system shown in Figure 6.10. It is an instance of the architecture presented in Figure 6.5. The forward and backward projections between the input and hidden population are here identical in both directions and also multiplexed, meaning that all receptive fields share a single copy of the forward-backward projection. We use this system to store hundred true color 128×128 pixels images taken from the COIL-100 database shown in Figure 6.12 (Nene, Nayar et al. 1996).

The purpose of the forward projection is to make the coding more compact instead of making it sparse as in section 6.3.1. In the input population each pixel is represented by three hypercolumns with 256 units in each. These three hypercolumns encode the pixels' RGB-values. The hidden population is composed of 16384 hypercolumns with 100 units in each. These 100 units code for hundred different colors that were computed by the CSL algorithm trained on all of the 16384 pixels in all of the hundred images.

The most computationally intensive part of this system was the recurrent projection of the hidden population. This projection implemented a giant attractor network with $1.6 \cdot 10^6$ units and $1.6 \cdot 10^{11}$ connections, to our knowledge one of the largest ever implemented and run on a cluster computer. Figure 6.11 shows the successive retrieval of image number 9 in the COIL-100 database.

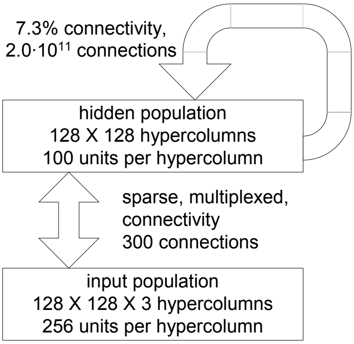


Figure 6.10. The memory system used to store the images from the COIL-100 repository.

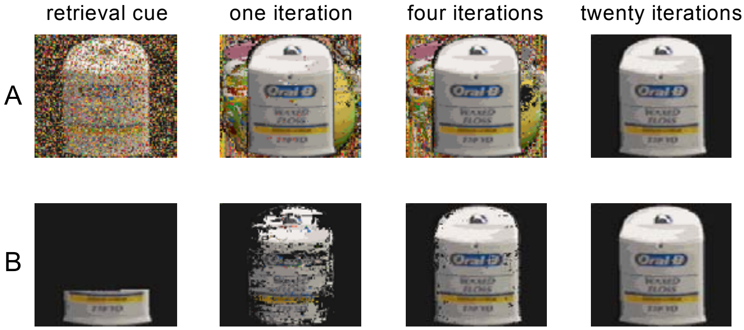


Figure 6.11. The successive retrieval of image number 9 in the COIL-100 database. The retrieval cues are shown in the left most column and the completely retrieved images in the right most column. The upper row (A) shows noise reduction from a retrieval cue with 70% noise. The lower row (B) visualizes pattern completion from a 30% partial view of the stored image.

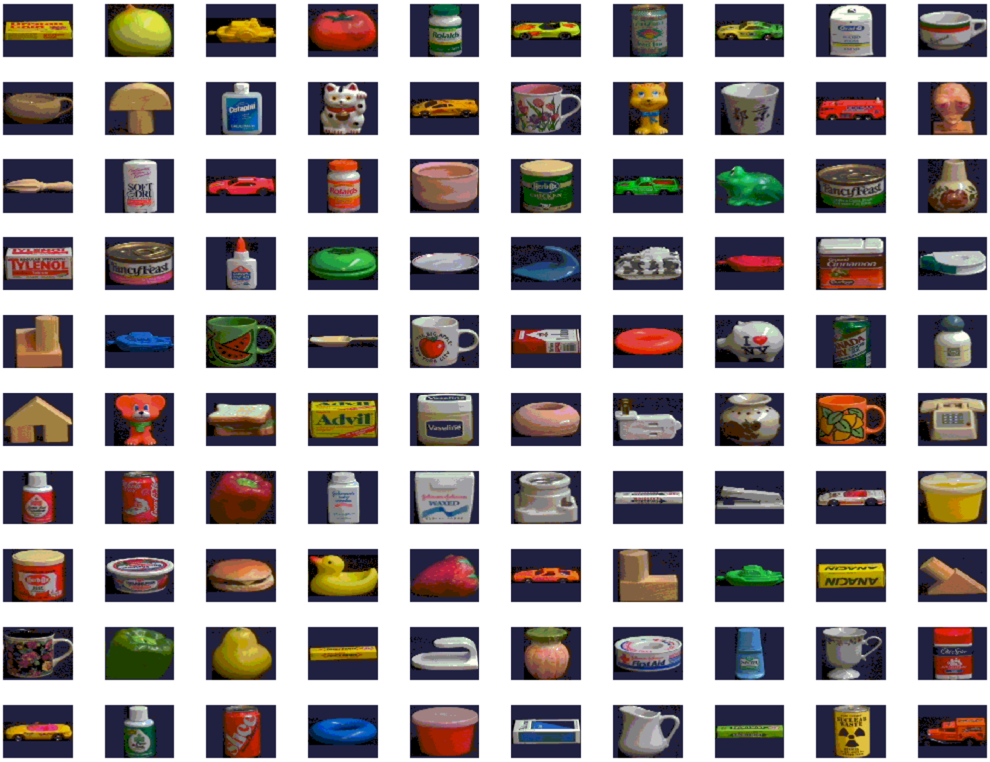


Figure 6.12. The COIL-100 image repository.

6.4 Classification with the Hierarchical Cortex Model

In this section we use a system based on the hierarchical cortical model for classification. These results have not been presented elsewhere. This system, shown in Figure 6.13, differ in a number of ways from the one used in section 6.3. The hidden population does not have a recurrent projection and it does not project back to the input population but instead it projects to an output population. The output population consists of a single hypercolumn and the units in this output population each represent a class label. Similar to the system in section 6.3, the projection from the input to the hidden population is trained with CSL and the projection going out from the hidden population is trained with the BCPNN learning rule.

During training of this system the class labels must be known because they are required when training the projection from the hidden to the output population. If we want the system to self-organize and find these class labels in an unsupervised manner, the BCPNN learning-rule used to train the projection to the output population must be replaced by CSL.

In this setup the system is similar to a radial basis function network (RBF) (Haykin 1999). They both have a first decorrelating stage based on a density estimation procedure followed by a linear classifier. This analogy gives a hint of the system’s capabilities in its present form and we expect that future developments will improve on these, e.g. by introducing invariances to the feature extraction.

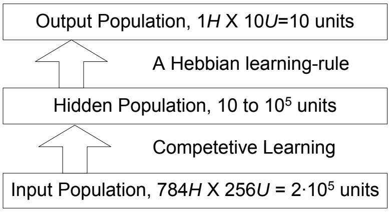


Figure 6.13. A classification system built on the framework of the hierarchical cortical.

To evaluate the performance of this classification system we used the MNIST data set (Lecun, Bottou et al. 1998), which contain 60000 training samples and 10000 test samples of handwritten digits between 0 and 9. In Figure 6.14 we have plotted the first 36 images of the training data set. Each image is represented with 28×28 pixels and 256 gray levels. LeCun et al. reports that support vector machines and other elaborate classifiers, with some tweaking, have a classification performance of 99% and above on the test data set.

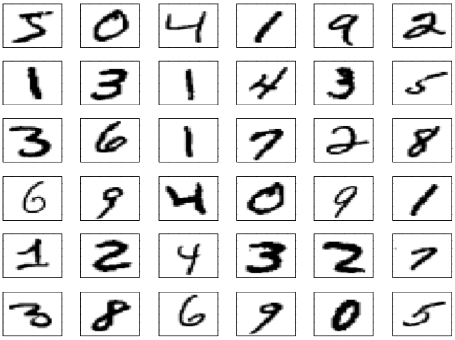


Figure 6.14. The first 36 images of the MNIST training data set.

First we used a single layer BCPNN, which is a linear classifier, to classify the handwritten digits. This classifier was constructed simply by removing the hidden population in Figure 6.13 and training the projection from the input to the output population using the BCPNN learning-rule. The results on the training and test set were 91% and 84% respectively. LeCun et al. 1998 report that they achieved 88% on the test set with a single layered network trained with a gradient descent type of algorithm and if a deskewing preprocessing step was used the linear classifier managed 91.6%.

Next we used the system with the hidden population for classifying the MNIST data. Three different organization principles for the receptive fields were tried; mutual information based clustering, random, and overlapping random fields. Figure 6.15 shows the partitioning generated by the mutual information based clustering that gave the best classification performance; 94.7% on the test data set. Here, the 28×28 pixels are divided into eight receptive fields. It is interesting to note that the rarely used pixels on the periphery of the images are clustered together. The fields also seem to have a tendency to stretch from the periphery towards the center of the image in elongated oval regions. When the number of areas is increased, many of the small receptive fields are located in the center of the input space, where they also are somewhat smaller than in the periphery.

In order to reduce the already heavy computational burden of the mutual information based clustering, the number of gray levels was reduced from 256 to 8 before the partitioning was computed. All of the 60000 patterns in the training data set were used when computing the clustering as well as training the projection to the output population.

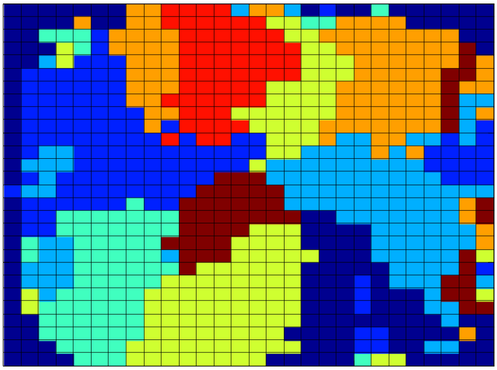


Figure 6.15. Partitioning of the input space (pixels) into eight color-coded receptive fields. This partitioning was generated by the mutual information based clustering algorithm (see section 6.2.2).

Figure 6.16 shows the classification performance plotted as a function of the number of code vectors in each receptive field. The results are plotted for four different numbers of receptive fields; 4, 8, 16, and 32. When the number of code vectors is large, the number of receptive fields is irrelevant for the classification performance, but when there are few code vectors the performance increases with the number of receptive fields.

In Figure 6.17 the classification performance is plotted as a function of the size of the hypercolumns in the hidden population, i.e. the number of code vectors in each receptive field. The figure also shows a comparison between using randomly formed receptive fields with and without overlap. In the case of overlapping receptive fields, each pixel had four times overlap. From these results we can conclude that allowing the receptive fields to overlap increases performance. Also, forming the receptive fields in an informed way improves the classification performance.

In order to test the quality of the representation created in the hidden population we did an experiment where the BCPNN projection from the hidden to the output population was replaced by a support vector machine (SVM). This was done for the system where the hidden population had H=8 and U=256, i.e. the input was divided into 8 receptive fields, each having 256 feature detectors. We experimented both with polynomial kernels of different degrees and radial basis function kernels, and found a polynomial kernel of degree 2 to produce the best results. Using this kernel, the classification performance was increased from 94.7%, when BCPNN was used, to 96.3%. This indicates that the feature detectors we used to represent the data, i.e. the competitive units, truncated the data slightly. It should be pointed out that these feature detectors were learnt using unsupervised training. If we instead trained this SVM directly on the raw MNIST data, the classification performance was 98.1%.

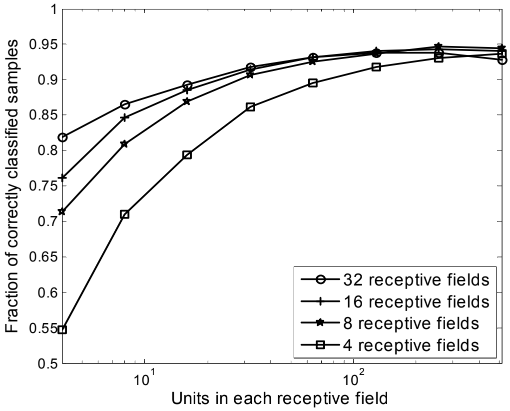


Figure 6.16. Classification performance plotted for four systems each with a different number of non-overlapping receptive fields. The number of hypercolumns, *H*, in the hidden population is equal to the number of receptive fields and the number of units in each hypercolumn, *U*, is varied on the x-axis.

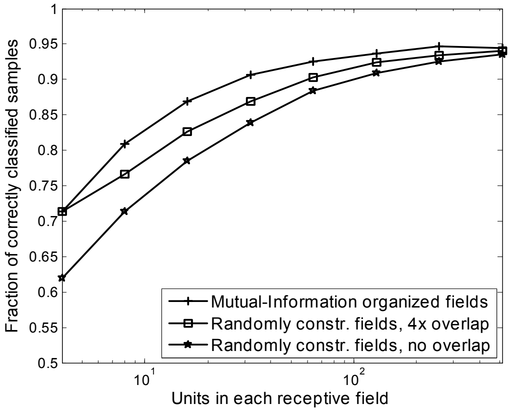


Figure 6.17. Classification performance plotted for three different types of receptive fields. There are 8 receptive fields in total for the cases of mutual information and randomly organized fields. In case of the randomly organized fields with overlap there are 32 receptive fields.

6.4.1 Using Unlabeled Data to Improve Classification Performance

Next we experiment with using unlabeled data to improve the classification performance. In many classification problems, there is typically a lot of unlabeled data available and the labeled data is scarce. The cost of producing labeled data is also typically large. Most of the machine learning methods used to create classifiers rely on the existence of an extensive set of labeled training data. Here we investigate if unlabeled data can be used to enhance the classification performance of our hierarchical system. The idea is to use both the labeled and unlabeled data with the CSL algorithm to train good feature detectors. Only the labeled data is then used to train the projection from the hidden to the output population. We compare the results of our classifier with those of a SVM that was only trained on the labeled data.

In Figure 6.18 the classification performance is plotted as a function of the number of labeled training data samples. The results for two different instances of our classifier based on the hierarchical cortical system are shown. The instance labeled “System 1” had a hidden population with *H*=8 and *U*=256, i.e. the input was partitioned into 8 receptive fields with 256 feature detectors in each. The other instance labeled “System 2” had a hidden population with *H*=8 and *U*=16. The support vector machine (SVM) had a polynomial kernel of degree 2. We experimented with other degrees of polynomials and also radial basis function kernels, but found the present kernel to be best.

The results show that the SVM outperform our classifier, also when our classifier is trained using unlabeled data. They also show that using fewer feature detectors, as in system 2, increases the classification performance when there are few labeled training data and using unlabeled data to train the feature detectors only marginally enhances the performance. From these results we conclude that SVM:s are very good classifiers, and it may not be relevant for a biological cognitive system, and in turn also a model of such a system, to have the same high classification accuracy as an SVM. For a biological cognitive system things like error tolerance and redundancy, speed, and the capability of unsupervised training may be more important than high classification accuracy.

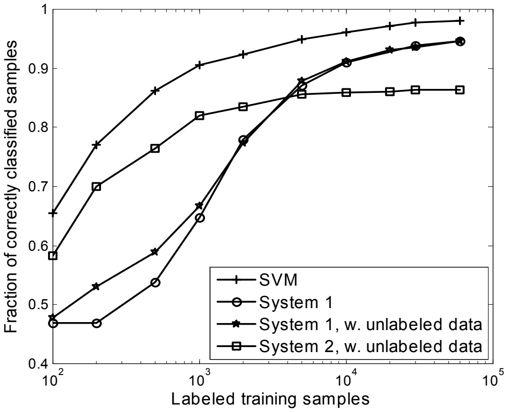


Figure 6.18. Classification performance plotted as a function of the number of labeled training samples. We have used both the classifier based on the hierarchical cortical model and a support vector machine (SVM). Two differently set up instances of our classifier is run, “system 1” with a hidden population that has *H*=8 and *U*=256, and “system 2” with a hidden population that has *H*=8 and *U*=16. System 1 and 2 were also tested using unlabeled data for training the feature detectors in our classifier.

The approach to using unlabeled data that is taken here is similar to that of Dara et al. who have constructed a classifier by combining a self-organizing map with a multi-layer perceptron trained with the Back-propagation algorithm (Dara, Kremer et al. 2002). The self-organizing map is trained using all available data, both labeled and unlabeled, and the multi-layer perceptron is trained using only the labeled data. In a paper by Nigam et al another approach is taken to using unlabeled data in training (Nigam, Mccallum et al. 2000). They use two classifiers that they first train using the labeled data. These classifiers are then used to label some of the unlabeled data. Then the classifiers are trained once more using the new extended data set of labeled data. This is then iteratively continued until all data has been labeled and the output from both classifiers agrees.

Dara et al. (2002) state that it is important for the data to have a topology that matches that of the feature extraction algorithm. Competitive learning, for example, assumes that data is grouped into spherical clusters in a multidimensional space and this might not always be the best way to describe the data.

7 Parallel Implementation of BCPNN

In this chapter we present and discuss a parallel implementation of the BCPNN based on its modular hypercolumnar structure. Results on running cortical sized networks on parallel cluster computers are presented. Further, we present running times on an instance of today's fastest computer in the world; Blue Gene/L. We also investigate an unorthodox approach to high performance computing on a commodity cluster.

Studying parallel implementations of abstract models of cortex will not only allow us to run larger systems faster than before, but it can also lead to a better understanding of how and why neocortex is designed as it is (Nelson and Bower 1990).

The issues discussed in this chapter are most relevant to the work in appendix I.

7.1 Parallelism in Neural Networks

Computational parallelism is present on several levels in a neural network and here we discuss how it can be exploited in implementations on cluster computers. In the following we list the major levels of parallelism in a neural network (Nordström and Svensson 1992; Hammarlund 1996; Serbedzija 1996):

- Weight parallelism – Parallelization over weights.
- Unit parallelism – Parallelization over units.
- Layer parallelism – Parallelization over groups of units.

In order to get good performance on a parallel computer it is crucial to match the computational grain size to the hardware and for a cluster with a few hundred nodes running a cortex sized network it means that layer parallelism should be exploited. Unit and weight parallelism could possibly be used locally on each node in the cluster. In BCPNN the layer type of parallelism is present in its hypercolumnar structure.

There are also two other kinds of parallelism associated with the learning phase present in the typical neural network:

- Training-session parallelism – Concurrent versions of a neural network are run.
- Training-example parallelism – Batches of patterns are simultaneously learnt.

Training-session parallelism means that several identical networks are run simultaneously e.g. with different parameters or slightly different inputs. This type parallelism is useful when doing parameter searches. It is also used when performing optimization with an attractor network in which case the final solution is selected from a couple of candidate solutions. Training-example parallelism can be used for some algorithms that implements batch learning, i.e. algorithms that train on a number of patterns simultaneously. In this thesis we focus on running a single large attractor network with incremental learning and therefore neither of these two types of parallelism are interesting to us.

7.2 Parallel Implementations of Neural Networks

This review focuses on recently developed implementations on cluster computers but we start off by giving a few selected references to older work done on e.g. the Connection Machine (Ghosh and Hwang 1989; Nordström and Svensson 1992; Levin 1995; Hammarlund 1996). A recent review of massively parallel computer hardware for running large neural networks is provided by Seiffert (Seiffert 2004).

The company Artificial Development has announced an architecture called CCortex that is aimed at simulating the entire brain on a cluster built of regular desktop computers. Artificial Development made a press release 2003 stating that they had run a neural network with $2{\cdot}10^9$ connections on 10^3 processors (CCortex 2003). The connections are represented with 8 bits and implement Hebbian learning.

SpikeNET is another framework for simulating large scale spiking neural networks (Delorme, Gautrais et al. 1999; Delorme and Thorpe 2003). SpikeNET can simulate millions of units and this is achieved by weight sharing, which means that the number of unique connections in the system is small. These connections are generally implemented without the ability to learn, i.e. adjust their synaptic strength.

At the Oregon Health and Science University, USA, Dan Hammerstrom is investigating how multiple specialized neural networks can be integrated into a larger modular, brain-inspired, system. His group is studying both hardware and cluster computer implementations of neural networks. In particular they focus on Willshaw-Palm networks with binary connection weights and they have run networks with up to 10^6 units (Zhu and Hammerstrom 2002).

The NEST (NEural Simulation Technology) initiative aims at developing a framework for simulating large-scale biologically realistic networks of spiking neurons. In a design proposal (Morrison, Mehring et al. 2005), techniques for designing memory efficient and scalable implementations are discussed. One technique used for making the communication in the simulation more efficient is to collect several messages in one package before communicating it. This is possible because real neurons have a communication delay due to the spike propagation in the axon. This project does not have the ambition to develop the neuron models, but merely the infrastructure needed for a parallel implementation.

At the university of Nevada, Reno, USA, there is a project aimed at building a framework for large-scale biologically realistic simulations of spiking units called NeoCortical Simulator (NCS). In a report from this project it has been calculated that it would be possible to run networks with up to $4{\cdot}10^8$ connections on a 128 processors cluster (Harris, Baurick et al. 2002).

The most ambitious project to large scale models of cortex is the Blue Brain project (Markram 2006). It is a collaboration between IBM, USA and Brain and Mind institute, EPFL, Switzerland that aims at creating a detailed simulation of individual neurons in the mammalian cortex. As a sidetrack, an abstract model with integrate-and-fire type of neurons is also developed, called the Blue Neocortex.

The currently largest simulations of biologically detailed conductance-based compartment models have been done using the SPLIT-simulator (Djurfeldt, Johansson et al. 2005). Simulations with up to 8 million neurons and 4 billion synapses have been performed.

We also want to leave a pointer to work done on implementing neural networks using the SIMD type of parallelism found in standard desktop processors (Strey 2003).

7.3 Parallel Computers

Parallel computers are typically classified into two categories; *SIMD* (single instruction, multiple data) and *MIMD* (multiple instructions, multiple data) type of parallel computers. The former type usually has a huge number (10^4) of very simple processors that all execute the same instructions i.e. program. The second type of parallel computers are usually built with a moderate number (10^2) of general-purpose processors that all execute a separate program. Within the second class of computers a distinction is also made between those that have a common memory for all nodes, *shared memory*, and those in which the memory is local and distributed over all nodes. MIMD type of computers with distributed memory are programmed according to the SPMD (single program, multiple data) paradigm. The communication between the nodes is explicitly included in the programs by calls to a communication API (e.g. MPI or PVM, see section 7.3.1), which is referred to as *message passing*.

The SIMD type of computers was dominating the scene up until the mid 1990's before the MIMD class of computers took over. Maybe the most famous computer of the SIMD class was the line of Cray supercomputers. Another computer of the same class, which many different types of neural networks has been implemented on, is the Connection Machine (CM) (Levin 1995; Serbedzija 1996; Misra 1997; Sundararajan and Saratchandran 1998). For the technically oriented reader we note that the Cray computers are vector machines while the CM has a distributed architecture. The CM had a large number of nodes interconnected by a hypercube network. In the typical implementation of a neural network on this machine, a single unit is mapped onto each node. As the number of nodes typically was very large, the communication usually became a bottleneck in these implementations.

In this thesis we have worked with *cluster computers*. They belong to the MIMD category and can be built by connecting more or less standard desktop computers together into a parallel computer. They typically have a distributed memory architecture, but by adding a software abstraction layer they can also be programmed as shared memory computers. In parallel computers, three levels of parallelism are present (Hammarlund 1996): The first level of parallelism is that of executing multiple instructions simultaneously on a single processor. This can either be done by pipelining (explained below) or by using several arithmetic units as in a SIMD computer. The second level of parallelism is that between several processors on one node. The third level of parallelism is that between nodes. This means that in the typical cluster computer, parallelism is present at several levels and, similarly, applications should be parallelized on multiple levels in order to fully utilize the capacity of the cluster.

Pipelining is a method that allows an arithmetic unit to perform several computations simultaneously. This means that the arithmetic unit can come close to producing one computation per clock cycle, although an operation such as e.g. multiplication takes on the order of tenths of clock cycles.

Implementing programs and achieving good performance on parallel computers is generally complicated (Hammarlund 1996). Many aspects such as the memory structure and inter node communication bandwidth, memory bandwidth, caches, processor architectures, node-network topology, buffers and communication protocols have to be considered. In the case of a heterogeneous cluster, achieving a good load balancing is also necessary but difficult (Grassmann and Anlauf 1999).

7.3.1 Message Passing

The standard of parallel programming on cluster computers is to use message passing. The two most commonly used API:s (application programming interface) are; MPI (Message Passing Interface) (Forum 1995) and PVM (Parallel Virtual Machine) (Geist, Beguelin et al. 1994). Both of these two API:s have bindings to Fortran, C, and C++ as standard, and bindings to Java have also been developed, see (Johansson and Lansner 2003).

MPI was designed with the intention of providing a standard for message passing and before its introduction all vendors of parallel computers had provided their own API for message passing. MPI has proved very successful and it has been widely adopted. In order to achieve the best performance and a maximum utilization of the interconnect, vendors of cluster computers and interconnects usually provide a specially designed version of MPI that is tweaked for their particular hardware (Luecke, Raffin et al. 2000). There is also a free-ware version of MPI, MPICH (Forum 2003), that implements the API on clusters and LAN:s (local area network). MPICH uses TCP/IP as its default communication protocol and it typically has a lower performance than hardware-vendor developed versions of MPI.

PVM was designed with the aim of providing a message passing API that could be used on heterogeneous networks of computers. The functionality in PVM is generally simpler than in MPI and there exits no support for optimizations that utilizes heuristics about the particular communication topology of a cluster. PVM is, similar to MPICH, usually implemented with the TCP/IP communication protocol.

7.3.2 Lenngren

Most of the work in this thesis was done on Lenngren. Lenngren is a Dell Xeon cluster at the Center for Parallel Computers (PDC), Royal Institute of Technology, Stockholm, Sweden. Lenngren has 442 nodes, each equipped with two Xeon processors running at 3.4 GHz and sharing 8 GB of memory. Each node has a peak performance of 13.6 GFLOP and a peak memory bandwidth of 6.4 GB/s. The nodes are connected with an Infiniband network and MPI is implemented with Scali MPI Connect (ScaMPI). The cluster is also equipped with a Gigabit Ethernet for file access and diagnostics.

7.3.3 Blue Gene/L

Blue Gene/L (BG/L) is the latest development in supercomputers from IBM. It represents a new breed of cluster computers where the number of processors, instead of the computational performance of individual processors, is the key to high performance (Gara, Blumrich et al. 2005). The power consumption of a processor and also the heat generated, increases quadratically with clock frequency, and therefore the amount of heat generated can be decreased significantly by lowering the clock frequency. Furthermore, this allows processors to be mounted more densely and less cooling equipment is needed for the computer. A system built in this way can be made to consume less material and less power for an equal computational performance, given that it can be fitted with a interconnect that has a high enough performance.

A BG/L can have up to 65536 processing nodes, with 2 processors per node. The largest simulations described in this thesis were performed on one BG/L rack containing 1024 nodes, which should be compared with a maximum of 256 nodes used on Lenngren. The theoretical peak performance of a processor in BG/L, running at 700 MHz, is 2.8 Gflops/s. This assumes using a special compiler that support the double floating point unit in the

PowerPC 440 core. Since we did not use that we instead count on a theoretical peak performance of 1.4 Gflops/s per processor.

A node in the BG/L cluster has 512 MB of memory compared to 8 GB for a node in the Lenngren cluster. A set of BG/L nodes can be configured to run in *co-processor mode*, where one of the two processors on a node run application code in most of the available memory while the other processor manages communication, and *virtual node mode*, where both processors run application code with half of the memory allocated to each. Thus, for our purposes, a Lenngren node has 5 times more computing power than a BG/L node in virtual node mode and 16 times more memory.

BG/L has five different inter node communication networks: a 3D torus network for point-to-point communication, a tree structured network for collective communication, a barrier network, a Gigabit Ethernet for file access, and a Fast Ethernet for diagnostics.

The BG/L computer used in this thesis is installed at the Deep Computing Capacity on Demand Center, IBM, Rochester, MN, USA. I personally did not have access to the machine and so I relayed on Mikeal Djurfeldt to run my programs, many thanks to him.

7.4 Implementation of BCPNN

In this section we present an efficient parallel implementation of the abstract cortical model based on a BCPNN. The BCPNN we consider is randomly and sparsely connected, which means that it is not possible to map the network in a clever way onto the nodes of the cluster in order maximize the utilization of their interconnect. One iteration of the network is composed of two phases, a learning and a retrieval phase. In the learning phase all connection weights are updated, not only those that are active in the current memory pattern. This is necessary for implementing the palimpsest property of the learning-rule. During the retrieval phase new activities are computed for all units.

In the following we consider the three different implementations; P-, Z-, and E-type, here listed in their order of complexity going from the least to the most complex network. Let Z be the number of connections per unit. Both the learning and retrieval phases scales as $O(HZF)$ for the Z- and P-type networks and in case of the E-type network the learning phase scales as $O(NZF)$. This means that the computational requirement of the proposed cortical model is to a large extent dependent upon the total number of connections. The memory requirement scales as $O(NZ)$ i.e. with the total number of connections.

Four techniques were used to achieve good performance (section 7.4.1-7.4.4); computation of the connection variables in the logarithmic domain, delayed updating of the connection variables, adjacency lists for effective indexing of the units in the sparse connectivity matrix, and AER for effective communication.

The networks were implemented using the programming language C with calls to two standard routines in the MPI API; MPI_Bcast() and MPI_Allgather().

7.4.1 Address Event Representation

The basic idea with address event representation (AER) is that the parallel interconnects (found in biological systems) are replaced with a single high-speed interconnection that use multiplexed coding of the units activities (Bailey and Hammerstrom 1988; Mortara and Vittoz 1994; Deiss, Douglas et al. 1999; Mattia and Giudice 2000; Culurciello and Andreou 2003). For a network with spiking units the required communication bandwidth is reduced to a minimum using AER. By using AER it is possible to run simulations with up to 10^6

units without suffering from a communication bottleneck on a commodity cluster computer with a standard Ethernet interconnect (Delorme, Gautrais et al. 1999). In an in-depth study by Culurciello and Andreou of AER, both the communication on-chip and off-chip is analyzed with respect to maximizing the throughput, reducing the latency, and power consumption (Culurciello and Andreou 2003). AER is sometimes referred to as spike lists. For more information on neural networks implemented with spiking activity see Schæfer et al.; (Schæfer, Schnauer et al. 2002).

7.4.2 Computations in the Logarithmic Domain

The update of the BCPNN connection-variables can be efficiently computed in the logarithmic domain. These state-variables are updated using EWMA:s and the network activity is typically sparse and unary coded, i.e. only one unit is active in each hypercolumn. Computing the EWMA:s in the logarithmic domain makes it more costly to have an input larger than zero, but an input equal to zero takes only two operations to compute and is therefore cheaper to compute than in the linear domain. Because most inputs are equal to zero there is a lot gained by performing the computations in the logarithmic domain. In case of the cortical model that has 1% activity, 99% of the units and 99.99% of the connections in each update can use the cheaper computation.

7.4.3 Delayed Updating

Delayed updating means that only the necessary work is performed in each time-step and the remaining computations are deferred to a later point in time. Delayed updating is similar to event driven updating in that only the necessary computations are performed at any particular moment (Delorme and Thorpe 2003; Mehrtash, Jung et al. 2003; Reutimann, Giugliano et al. 2003). In order to use delayed updating it is necessary that the BCPNN-variables are computed in the logarithmic domain. The value of an EWMA following several inputs equaling zero can then be extrapolated as a linear function of time which results in a very efficient program. In the implementation a time stamp is attached to all state-variables, and only when a variable is needed is its value computed and the time stamp updated. An implication of this is that some of the work associated with training the network and computing the weights is shifted to the retrieval phase.

There is also an indirect benefit from using event driven updating. Only a few connections are updated in each iteration and thus the memory bandwidth needed is very much reduced as compared to the case were all connections are accessed in each iteration.

7.4.4 Adjacency Lists

We use an adjacency list representation of the sparse connectivity. By using this type of memory structure, both updating the weights in the training phase and computing the support values in the retrieval phase can be done very efficiently.

The connectivity in the network is first setup so that each unit keeps a list of its presynaptic neighbors. All units in a hypercolumn can have different presynaptic neighbors. Then, in a second phase, the connection matrix is transformed for each hypercolumn into an adjacency list representation. The hypercolumn then gets a list for each presynaptic unit containing the postsynaptic units they connect to in that particular hypercolumn. The efficiency of this representation stems from that a connection is indexed by the presynaptic unit, and no search is needed to find out if a connection exits.

7.4.5 Three Different Implementations

Next we discuss the three different implementations of the learning phase, each with different computational complexity. In the simplest implementation, with the lowest complexity, the units' activity rates are computed by EWMA:s directly from the input as in Figure 5.3. Here, S_i is the input, P_i is the probability that a unit is active and P_{ij} the probability that two units are simultaneously active. This can be implemented with event driven updating of both unit and connection variables, which are computed in the logarithmic domain. We call this a *P-type* implementation. A connection need 10 bytes of memory; 4 bytes for storing P_{ij}, 4 bytes for the index of the presynaptic unit, and 2 bytes for the timestamp.

In a slightly more complex implementation, called *Z-type*, each unit is updated with two EWMA:s as in Figure 5.4. Here, we only use event driven updating of the connection variables. A connection in the Z-type network requires as much memory as in the P-type of network.

The implementation with the highest complexity is called *E-type*, and it uses two EWMA:s for computing each connection's variables and three EWMA:s for computing each unit's variables as shown in Figure 5.5. Here, it is not possible to use event driven updating and hence the training time increases considerably compared with the two simpler implementations (P- and Z-type of networks). In each iteration during training, all connections are updated. A connection need 12 bytes of memory; 4 bytes for storing P_{ij}, 4 bytes for storing E_{ij}, 4 bytes for the index of the presynaptic unit, and 2 bytes for the timestamp.

7.4.6 The Hypercolumn Module

The number of units and connections in a hypercolumn are constant. Therefore are the computational requirements of a hypercolumn invariant with respect to the size of the overall system given that its' units connects to a constant number of other units. This makes the hypercolumn a perfect module when building a large system on a cluster computer or in hardware. In particular when building a hardware implementation, having a module with a constant memory requirement that can be duplicated in order to increase the size of the neural system is advantageous.

A hypercolumn module includes U=100 units and Z=$1.2\cdot10^5$ connections per unit (assuming five synapses for each mini-to-minicolumn connection) and their respective variables, and it has ZU=$1.2\cdot10^5\cdot100$=$1.2\cdot10^7$ connections in total. Each hypercolumn also has a local set of variables that is used to compute the bias activity of the pre- and postsynaptic units. In the case of a floating-point arithmetic version of a P-type implementation each connection is stored with 10 bytes and 114 MB of memory is needed but in the case of a fixed-point arithmetic implementation where each connection is stored with 7 bytes only 80 MB is required. For real-time operation the average number of connections that have to be processed per second is ZF=$1.2\cdot10^7$. This requires a memory bandwidth to the processing unit capable of 115 MB/s and based on actual experiments the computing power needed has been estimated to a processor with a peak performance of 1.5 GFLOP.

7.5 Running the Cortex Model

In this section we present results on the scaling and running times of the cortical model. We study both implementations with different complexity and implementations done using both floating-point and fix-point arithmetic. These results are presented in appendix I and in a report by Djurfeldt et al. (Djurfeldt, Johansson et al. 2005).

Both of the two cluster computers that we used have two processors per node that both share a common memory bus. Therefore, when running on both processors simultaneously, their effective memory bandwidth is half that of the memory bus, which can have a negative impact on performance. But when running our code on both Lenngren and BG/L we did not see any significant difference in performance between a set of nodes running in single processor mode and dual processor mode.

The results in sections 7.5.1 and 7.5.2 are taken from the report (Djurfeldt, Johansson et al. 2005), and those in section 7.5.3 were presented in appendix I.

7.5.1 Comparison of the Three Different Implementations

Here we compare the running times of all three implementations of BCPNN, and we do this both on BG/L and Lenngren. We run a network with 2048 hypercolumns and $1.2 \cdot 10^5$ connections per unit. This network is 30% larger than a mouse cortex equivalent network (Johansson and Lansner 2004d; Johansson and Lansner 2004c). The programs were designed to fill the memory on each node used. Therefore, the simulations run on BG/L used 16 times more nodes than those run on Lenngren.

Table 7.1 shows the iteration times for 128 processors on Lenngren. On each processor 16 hypercolumns were allocated. Running the Z-type was 3.3 times slower and E-type was 72 times slower than P-type. Running the E-type was 22 times slower than Z-type.

Table 7.2 shows the iteration times for 2048 processors on BG/L. On each processor one hypercolumn was allocated. Running the Z-type was 3.6 times slower and E-type was 30 times slower than P-type. Running the E-type was 8 times slower than Z-type.

The retrieval times for the Z-type network were the longest because two logarithms are computed for each weight in the sum of synaptic input to each unit. In case of the P-type network no logarithm needs to be computed in the synaptic summation, and for the E-type one logarithm is computed for each weight in the synaptic summation.

Table 7.1. The iteration times for networks with three different levels of complexity run on 128 processors on Lenngren. Time is measured in seconds.

	Train	**Retrieval**	**Total**
P-type	0.0130	0.0196	0.0326
Z-type	0.0270	0.0825	0.109
E-type	2.29	0.0538	2.344

Table 7.2. The iteration times for networks with three different levels of complexity run on 2048 processors (16·128) on BG/L. Time is measured in seconds.

	Train	**Retrieval**	**Total**
P-type	0.007	0.0184	0.0254
Z-type	0.018	0.0736	0.0916
E-type	0.718	0.0415	0.756

As we discussed in section 7.3.3, a processor in Lenngren is approximately five times faster than one in BG/L. For the presented simulation results it means that the running times on BG/L should be 3.2 times shorter than on Lenngren. In Table 7.3 we have computed the ratios between the running times on BG/L and Lenngren. We find that the computationally very intensive E-type of network scales very well to the greater number of processors on BG/L and uses the computational resources in this computer as well as in Lenngren.

Table 7.3. Ratio of running times taken from Table 7.1 and Table 7.2 on Lenngren and BG/L. Theoretically, 2048 processors in BG/L run a program 3.2 times faster than 128 processors in Lenngren.

	Train	**Retrieval**	**Total**
P-type	1.9	1.1	1.3
Z-type	1.5	1.1	1.2
E-type	3.2	1.3	3.1

7.5.2 Scaling of the Three Different Implementations

As in the comparison of the two different implementations, the scaling experiments were setup to reflect the fact that a node in BG/L only has 1/16 of the memory compared to a node in Lenngren, in the way that 16 times more nodes were used to run the programs on BG/L. This means that the programs filled the memory on each node used, on both clusters. The scaling was studied under the condition that the problem size was increased together with the number of processors, which is referred to as *scaled speedup*. If the program parallelizes perfectly the scaled speedup measure is constant. We did not use the common *speedup* measure where the running time of a problem with constant size is plotted as a function of the number of processors. The reason being that the application studied has a relatively large and fixed resident part that must be implemented on each processor. When the problem size is scaled more than an order of magnitude, this resident part becomes very influential on the running times for the smaller sized networks. Neither did we plot the scaling of the parallel program relative to an optimal implementation on a single processor. This is because the problem is very memory intensive and it is not possible to run this type of networks on a single processor. We need to run the networks on cluster computers not only for the processing power but also for the available memory resources.

The problem size was varied in two ways; Firstly, the number of units was increased, each having a constant number of connections. Secondly, the number of connections per unit was increased, holding the number of units constant.

In Figure 7.1 the processing time per iteration is plotted as a function of an increasing network size for both Lenngren and BG/L. The number of processors used to run the networks was increased simultaneously with the networks' sizes, meaning that the number of hypercolumns per processor was constant. The number of connections per unit was also fixed to $4{\cdot}10^4$. In these plots, perfect scaling is achieved if the iteration times are constant. We found that the retrieval phase scaled better (constant iteration times) than the training phase, which had slightly increasing iteration times.

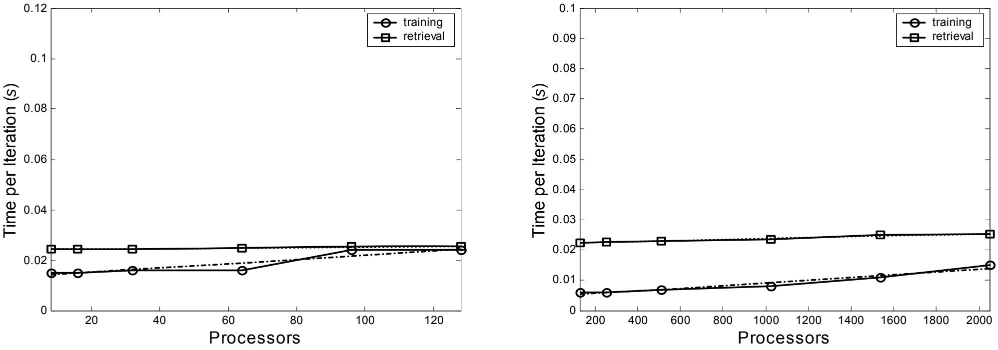


Figure 7.1. Processing time per iteration plotted as a function of an increasing network size and an increasing number of processors for a P-type network. The number of hypercolumns per processor was constant and the number of connections per unit was fixed to 40000. (Left) Results on Lenngren for networks with 64 hypercolumns per processor. (Right) Iteration times on BG/L for networks with 4 hypercolumns per processor.

In Figure 7.2, left, the scaling, on Lenngren, of a P-type network when the number of connections per unit was increased is plotted. The number of hypercolumns was fixed to 2048. Figure 7.2, right, show the iteration times on BG/L, where 16 times more processors were used than on Lenngren. From these results we can conclude that neither the training nor the retrieval times scale faster than the increase in processing power when more connections are added, i.e. the network scales perfectly.

For more results on the scaling see Djurfeldt et al (2006).

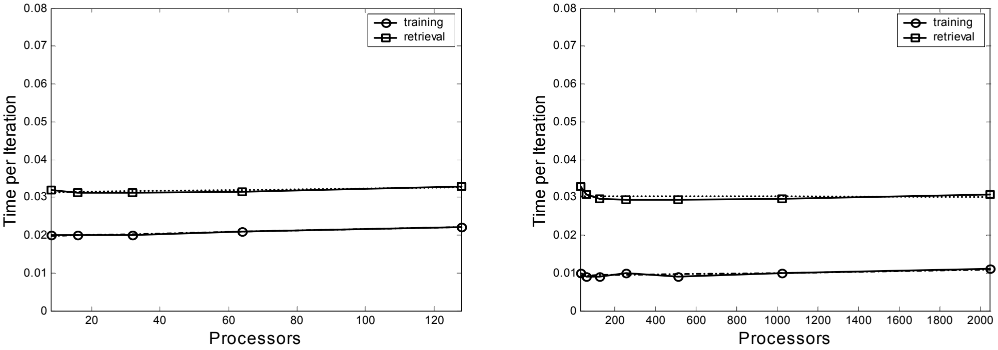


Figure 7.2. The iteration times on Lenngren (left) and BG/L (right) for a P-type network with a constant number of 2048 hypercolumns.

7.5.3 Running Times for the Cortical Model

In this section we present the running times on Lenngren for both floating- and fixed-point arithmetic (see section 8.3) implementations of a P-type network. In the following we refer to a mouse-sized instance of the cortical model as a M-sized network, a rat-sized instance as R-sized, a cat-sized instance as C-sized, a macaque-sized instance as Ma-sized, and a human-sized instance as a H-sized network. We also discuss an I-sized network with a size between that of a R- and C-sized network. In Table 7.4 running times on 256 nodes and 512 processors on Lenngren are presented. The largest network we ran, referred to as the I-sized network, had 16384 hypercolumns, $1.6\cdot10^6$ units, and $2.0\cdot10^{11}$ connections. The floating-point arithmetic version of this network allocated 7.2 GB of memory on each node and a total of 1844 GB memory. The M-, R-, and I-sized networks implemented with floating-point arithmetic run in 42%, 22%, and 8% of real-time respectively.

The fixed-point arithmetic version of the algorithm was slightly faster than the floating-point arithmetic implementation. Further, it used 30% less memory than the floating-point arithmetic version.

Table 7.4. Memory usage and running times for both floating- and fixed-point arithmetic implementations of a P-type network. FA – Floating-Point Arithmetic IA – Fixed-Point Arithmetic.

	M-sized FA	**M-sized IA**	**R-sized FA**	**R-sized IA**	**I-sized FA**	**I-sized IA**
Units	$1.5\cdot10^5$	$1.5\cdot10^5$	$5.1\cdot10^5$	$5.1\cdot10^5$	$1.6\cdot10^6$	$1.6\cdot10^6$
Connections	$2.4\cdot10^{10}$	$2.4\cdot10^{10}$	$6.1\cdot10^{10}$	$6.1\cdot10^{10}$	$2.0\cdot10^{11}$	$2.0\cdot10^{11}$
Bytes per Connection	10	7	10	7	10	7
Total Memory (GB)	221	155	576	405	1844	1295
Memory per Node (GB)	0.9	0.6	2.3	1.6	7.2	5.1
Updating Weights (ms)	9	9	19	16	51	47
Updating Activities (ms)	14	12	25	21	61	59

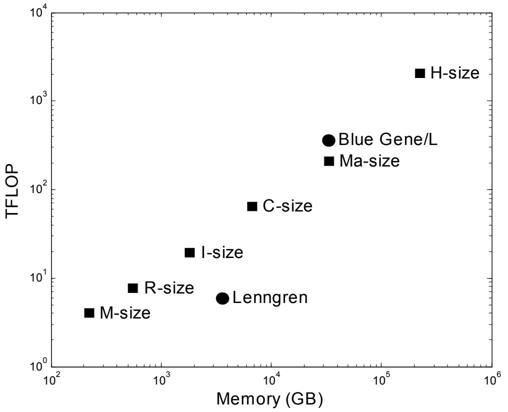


Figure 7.3. The peak performance of Lenngren and BG/L related to the requirements for real-time operation of the cortical model. Six differently sized instances of the floating-point implementation of the model are plotted.

In Figure 7.3 the computational requirements, in terms of memory usage and processor peak performance, for real-time operation of M-, R-, I-, C-, Ma-, and H-sized systems are estimated. These estimates are based on the time spent on computations and do not include the time required for communication. Also shown in Figure 7.3 is the performance currently available on the full Lenngren cluster and the full BG/L machine. This plot shows that a Ma-sized network almost fit in the memory of BG/L and the computer's computational performance is sufficient to run the network in real-time. It also suggests that the M-sized network could run in real-time on the full Lenngren cluster.

A testimony of the code's efficiency comes from setting the unofficial record in power use on Lenngren when running the I-sized network. Lenngren actually used more power to run this network than it used when running the benchmark programs for the TOP-500 supercomputers list. The power usage peaked at around 120 kW, which meant that the average connection drew $6 \cdot 10^{-7}$ W. Translated into the biological equivalent, this is roughly 10^{-8} W per synapse. The corresponding figure for the human brain is $3 \cdot 10^{-14}$ W per synapse given that the human body consumes energy at a rate of 150 W (Henry 2005) and that 20% of this energy is used by the 10^{11} neurons and 10^{15} synapses of the brain (Kandel, Schwartz et al. 2000).

7.6 Communication

In this section we discuss the communication requirements of the cortical model when run on Lenngren and BG/L. We also present an alternative communication solution that can be used to achieve fast communication when running a neural network on a low performance Ethernet.

The speed of an interconnect is determined by two factors; latency and bandwidth. In the case of global broadcast communication, the speed is also very dependent on the interconnect's structure and the message size. How to implement the fastest broadcast communication depends on these two parameters as well as the latency and bandwidth.

If the computational dependencies in a program can be matched with the topology of the interconnect it is possible to get a very efficient communication solution. A good example of this is a special type of neural networks, called cellular neural networks (Chua and Yang 1988), that have been developed to map well onto computers with a grid based interconnect. In these neural networks, a unit only takes inputs from its neighbors as defined by the topology of the interconnection grid, thus enabling a high performance.

In case of the cortical model instantiated by a randomly connected attractor network there is no structure in the connectivity. In case of a network with random sparse connectivity it is not possible to optimize the communication by using local transfers, instead all-to-all communication has to be used. We have previously concluded that the inter-hypercolumn communication can be multiplexed by AER (section 7.4.1) and that the required bandwidth is relatively low even for cortically sized networks (Johansson and Lansner 2004b). In appendix I we show that it is possible to use all-to-all communication when running cortical sized attractor networks and that these implementations are bounded by computations and not communication.

Implementing a communication scheme in which only units that change their activity communicate as opposed to the current implementation where all active units communicate could further reduce the communication requirements in the cortical model (Jonsson 2003). In the case of more sophisticated implementations of the cortical model, where groups of hypercolumns (cortical areas) are connected hierarchically, it is a good idea to exploit this topology to implement efficient communication when mapping it onto a cluster computer.

As discussed earlier in section 7.3, parallelism exists on several levels in a typical cluster computer and together with it comes a communication hierarchy. Typically, the lowest levels in this hierarchy have the smallest latencies. Also stated previously, trying to optimize a program for all these levels of parallelism that each has different communication premises is difficult.

7.6.1 Communication Networks in Cluster Computers

Interconnects in cluster computers can be divided into four main categories (Ghosh and Hwang 1989); dynamic (switched), direct, bus based, and multistage (grid) networks. In the following we discuss each of these types. All of the networks we consider here use package-based communication.

Ethernet and Infiniband are examples of switched networks. If all nodes in the network can communicate simultaneously at the maximum bandwidth, we say that the network, and the switch, has a full bisectional bandwidth. Typically, the switch does not have the capability to support simultaneous communication between all nodes. In this case, a number of nodes share a single communication channel to the switch, which then works as a bus based interconnect. Typically, a switch only has 16 ports. But by connecting switches in a fat-tree, a large number of switches can be aggregated to form an abstract switch with an arbitrary number of ports. In case of Ethernet it is not possible to aggregate switches efficiently and handling the TCP/IP stack consumes a lot a processing power on each node (Bode, Hill et al. 2004). This makes Ethernet ill-suited as an interconnect in a high-performance cluster (Vaughan 2003).

An example of a network with direct connectivity is Myrinet, in which a crossbar switch connects the nodes. As opposed to the switched networks, the switch does not need to route packages. Similarly to switched networks, this type of network can also be characterized by its bisection bandwidth.

Here we mention bus based networks because Ethernet, for example, uses this kind of communication topology on a local level. In a bus-based network a number of nodes are connected to a single communication channel and only one node at a time can send data onto the network. Collective communication can be effectively implemented on this type of network by broadcasting, where one node sends data to all other nodes simultaneously.

A multistage or grid based interconnect has nodes with a very limited number of connections to its closest neighbors. Typical a network of this type has a 1-3D topology. Examples of 1D topologies are the line and ring, in 2D we have the grid and torus, and in 3D and higher dimensions the hypercube is a common topology. This type of interconnect is also known as systolic arrays (Kung and Hwang 1988; Margaritis 1995; Mahapatra and Mahapatra 2000), in which the communication is implemented by circulation. In the case of a fully or randomly connected neural network implemented on a grid, the grid's topology only have a small, constant effect on the iteration time (Murre 1993). This means that going from a 1D to a 2D or 3D design only increases the performance with a small constant factor. In appendix I we calculate the time for an all-to-all communication using BG/L's grid network.

7.6.2 Broadcast Communication over Ethernet

In this section we investigate how to optimize global communication, such as all-to-all broadcasts, in a cluster with a low performance Ethernet interconnect by using the UDP/IP multicast. This work has previously been presented in a report (Johansson and Lansner 2003). The UDP/IP multicast does not have any error control and hence there can be missing packages and packages arriving out of order. But in the case of neural network implementations it is often not necessary to have error free and synchronized communication, these faults can typically be absorbed by the network itself (Jonsson 2003). A similar approach has previously been studied under the context of error-free communication with the aim of making global communication in MPI implementations on Ethernet more effective (Chen, Carrasco et al. 2000). We define a low performance Ethernet interconnect as one where the number of nodes outnumber the number of ports on all switches. We compare three different implementations; two in Java that use custom built implementations of the TCP/IP and UDP/IP communication, and one in C that uses MPICH.

Ethernet supports packaged based communication with IP (Internet Protocol). Two communication protocols that are commonly used together with IP are; TCP (Transfer Control Protocol) and UDP (User Datagram Protocol). When TCP/IP based communication is used, no messages are lost and the ordering of the data transfers is deterministic. Each communication event is preceded by a three-way handshake. Creating a communication channel in this way guarantees successful communication but the handshake takes time. This type of handshake is not used in UDP/IP based communication, which does not guarantee that messages arrive and there is neither any control of when messages arrive or in what order they arrive. The benefit of UDP/IP is a very small communication overhead and the support for very efficient broadcasts.

Next we present an implementation of all-to-all communication using both TCP/IP and UDP/IP, and we also present experimental performance results. In both implementations, one of the nodes is declared *server* while the other nodes are declared *clients*. The server is used to control the execution of the program.

In the TCP/IP based implementation of the ALL_GATHER transfer, all communication goes through the server (Figure 7.4). The server sends a data vector to all clients and the clients reply with their part of the data vector, which they have updated. This centralized communication scheme incurs a large communication overhead.

For the C implementation using MPICH, the communication scheme is identical to that in the Java/TCP implementation. But here, the latencies involved in each communication event are much shorter.

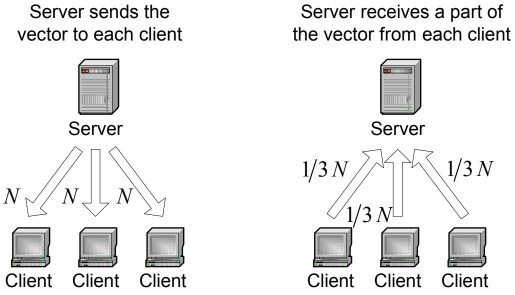


Figure 7.4. The communication scheme used to implement the MPI ALL_GATHER operation in the TCP/IP and MPICH based implementations.

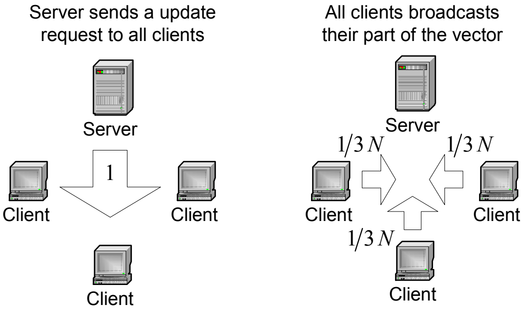


Figure 7.5. The communication scheme used to implement the MPI ALL_GATHER by the UDP/IP based implementation.

In the UDP/IP based implementation, the clients communicate directly with eachother (Figure 7.5). An update of the data vector starts with a signal from the server to all clients. The clients respond to this signal by computing and broadcasting their part of the vector. A problem that arises in the communication is that of collisions which results in lost packages. Therefore each client has an individually set delay after which it sends its activity vector.

The required bandwidth for a TCP/IP based implementation of this ALL_GATHER communication, B_{TCP}, is computed with eq. (7.1) and for UDP/IP, B_{UDP}, with eq. (7.2). Here, *data* is the size of the data vector, *header* is the size of a message-header, *nodes* is the total number of nodes, and *f* is the frequency of updates.

$$B_{TCP} = nodes(data + data / nodes + 2header) f \tag{7.1}$$

$$B_{UDP} = nodes(data / nodes + header) f \tag{7.2}$$

The scaling of UDP based communication was investigated on the SBC cluster located at PDC, using up to 64 nodes (Johansson and Lansner 2003). In the SBC cluster, the nodes are connected with a 100 Megabit Ethernet, and each node is equipped with an Athlon XP processor running at 1400 MHz and 768 Mbytes of memory. The parameters for eq. (7.2) are: *header*=12 bytes, *data*=100·8/*nodes* bytes, and $nodes \in \{2,..,64\}$. These values were chosen to simulate the communication in a BCPNN with 100 hypercolumns that each transmits 8 bytes of data in each update. The efficiency of the bandwidth usage was computed using eq. (7.1) for the Java/TCP and MPICH implementations and eq. (7.2) for the Java/UDP implementation.

The fastest communication solution was that in Java/UDP (Figure 7.6, left). The fraction of lost packages was never higher than 1% when using the Java/UDP implementation. This shows that very efficient broadcast communication can be implemented using this technique. By studying Figure 7.6, right, we see that the MPICH-C implementation used the available bandwidth most efficiently. An interpretation of this is that the Java implementations had much longer communication latencies than the MPICH-C implementation.

For a BCPNN with *H* hypercolumns, the required bandwidth when Java/TCP is used scales as *O*(*H*·*nodes*) and as *O*(*H*) when Java/UDP is used.

The key to successfully using this type of non-error free hardware is the neural network's capability of absorbing the errors that occur. This capability is not unique to neural networks and all stable iterative algorithms should be possible to use with non-error free hardware.

A more efficient implementation of the Java/UDP communication has been investigated in a master thesis work (Shakorian 2005). In that thesis an implementation in C is studied and evaluated. It is concluded that all-to-all broadcast communication using a custom built C/UDP implementation is much slower than one using an optimized MPI implementation.

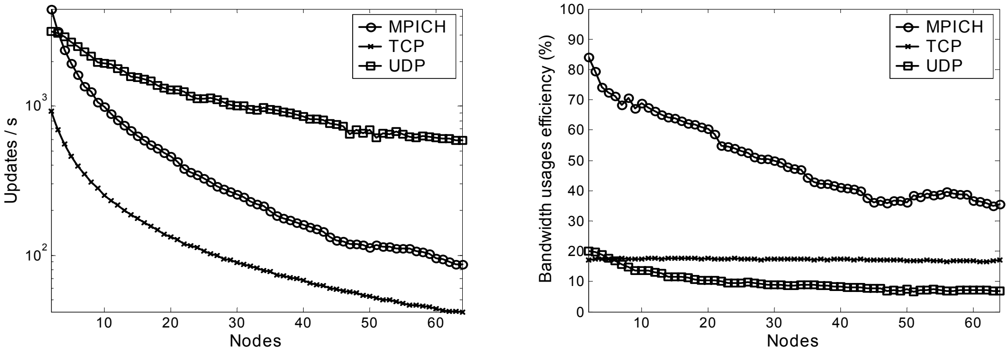


Figure 7.6. All-to-all broadcast communication on a 100 Megabit Ethernet. (Left) The number of completed updates per second for three different communication implementations. (Right) Percentage of the peak bandwidth utilized by the three different implementations. The MPICH implementation utilizes the bandwidth five times more efficiently than the UDP/IP implementation, but the UDP/IP implementation achieves a seven times higher update frequency. No more than 1% of the UDP/IP updates were incomplete. 100 hypercolumns were simulated; each hypercolumn generated 8 bytes of data on each update. A message's header contained 12 bytes. The scaling was studied from 2 to 64 nodes.

8 Hardware Designs of BCPNN

In this chapter we discuss custom built hardware for running neural networks with a focus on digital implementations. In order to achieve an efficient digital hardware design it is necessary that the algorithm of the intended design can be implemented with low-precision fix-point arithmetic. It is also possible to use floating-point arithmetic, but the costs in terms of transistors for both implementing the computations and storing the variables then increase considerably. In this chapter we propose a fixed-point arithmetic design of BCPNN and present an implementation of BCPNN on a programmable hardware named Mitrion.

This chapter presents the content of appendix V.

8.1 Custom Built Hardware for Running Neural Networks

To fully take advantage of neural networks' inherit parallelism custom-built hardware is needed. When designing custom-built hardware, several different approaches can be taken. The first and most important choice is that between using analog or digital circuits. In case of a system with multiple nodes there is the choice whether to ride piggyback on an existing computer network or to design a standalone system. Then there is also the choice whether to use custom designed chips such as FPGA:s (field programmable gate arrays) and ASIC:s (application specific integrated circuits) or to use general purpose processors and DSP:s (digital signaling processors). It is generally conceived that FPGA:s can increase the computational speed of an algorithm with as much as two orders of magnitude (Andersson, Johansson et al. 2006; Markram 2006). Further, moving from a FPGA to an ASIC can increase the speed by another two orders of magnitude. There are numerous papers that give an overview of custom built hardware made for running neural networks; (Nordström and Svensson 1992; Serbedzija 1996; Misra 1997), and there are also a number of introductions to the design of hardware neural network implementations; (Ghosh and Hwang 1989; Murre 1993; Mahapatra and Mahapatra 2000).

Most of the large-scale implementations of neural networks have been done using digital instead of analog hardware. With digital hardware it is easier to design large and complex systems. It allows for a greater flexibility in the design, e.g. fan-in and fan-out constraints are less severe and it is much more reliable. Further, digital hardware can be implemented with smaller circuit dimensions than analog hardware. It is also easier to reuse components and the design process is faster and cheaper. Finally, using digital hardware enables the use of a large number of existing building blocks known as IP-blocks (intellectual property).

The attracting features of analog circuits are that neural networks can be very efficiently implemented using very few transistors and they are also very energy efficient. The major weaknesses of analog designs are their inaccuracy, susceptibility to temperature fluctuations, and vulnerability to interference between transistors. But given that neural networks are inherently fault-tolerant, these obstacles may be overcome. For an introduction to analog implementations see Hopfield (Hopfield 1990) and for an overview of actual implementations see Draghici (Draghici 2000).

An interesting development is hybrid circuits that combine both analog and digital designs on the same chip. There have also been proposals of designs based on optical technology.

There have been numerous attempts to build hardware that is specially designed to run neural networks, but none of these have been really successful. The difficulty lies in building hardware that is faster than general-purpose parallel computers and at the same time works with a wide variety of neural network algorithms. Especially since general-purpose hardware as opposed to custom made hardware is typically fabricated using the latest silicon technology and thus follows Moore's law more closely.

The communication network in special-purpose built hardware is often physically implemented in 1-2D, but nerve cells are connected in 3D. This means that the brain has an advantage over electronic hardware in terms of connectivity structure. Implementing a large-scale neural network with hardwired connections in a 2D silicon design is not possible without multiplexed communication since the surface area required for the interconnections scales as $O(N^3)$ (Hammerstrom 1987).

8.1.1 Analog and Hybrid Designs

One of the most well known implementations of a neural network using analog technology is the commercial chip named ETANN (Electronically Trainable Analog Neural Network) made by Intel (Castro, Tam et al. 1993). This chip could hold 64 recurrently connected units and the weights were trained on a desktop host computer and then downloaded to the chip. It has been used in some experimental applications but because only small networks could be constructed and the training had to be done off-chip, it never became a big seller.

Analog circuits have also been used to design recurrent networks (Verleysen, Sirletti et al. 1989; Badoni, Bertazzoni et al. 1995; Giudice, Fusi et al. 1998; Fusi, Annunziato et al. 2000; Chicca, Badoni et al. 2003) and winner-take-all (WTA) networks (Indiveri 2001; Liu, Kramer et al. 2001) with spiking units. In particular, WTA circuits are simple to implement using analog transistors (Wuagh and Westervelt 1993a). We also give a reference to a hybrid design of integrate-and-fire neurons (Liu and Douglas 2004).

8.1.2 Digital Designs

The first step towards an efficient digital hardware implementation of an algorithm is to convert it into using fixed-point arithmetic and preferably also low precision variables (Draghici 2002). The algorithm can then be implemented with a large number of parallel computational units in the hardware and this is a requisite for high performance.

Typically the training algorithm requires weights with a higher precision than does the retrieval procedure. In an analysis of the required precision for the Back-propagation algorithm it is concluded that 7-15 bits are needed for the weights during training (Holt and Hwang 1993). The capability of a feed-forward network is reduced when the precision of the weights is decreased (Draghici 2002). In case of an implementation of the most basic autoassociative memory; the Willshaw-Palm network, only 1 bit per connection is required given that the network has full connectivity. In appendices I and V we propose a fixed-point arithmetic implementation of BCPNN that retain the same capabilities as those of a floating-point implementation. BCPNN is implemented with several bits per connection and it is more capable and versatile than a Willshaw-Palm network.

This thesis focuses on attractor networks and therefore we have in the following compiled a list of a few hardware implementations predominately of this type of networks. First we list a few designs made in ASIC:s, then FPGA:s, next DSP:s, and finally more general-purpose types of hardware. The list is far from exhaustive.

Johannet et al. describe the implementation of a Hopfield network with 64 units that implements learning (Johannet, Personnaz et al. 1992). It is concluded that 9-11 bits precision is necessary for a network with 64 units and dense activity. In a project that extends the work of Johannet et al., a Hopfield network with 1024 units is implemented (Hendrich 1996). The weights are stored with 8 bits precision in an off-chip SRAM memory. Another implementation of a Hopfield network with 1024 units has been proposed by Watanabe et al. (Watanabe, Kimura et al. 1993). The computations are done in 16 processing elements and the connections are stored in an off-chip DRAM with 8 bits precision. SP^{2}INN (Mehrtash, Jung et al. 2003) is a project aimed at building a dedicated neural processor in hardware that can run a large number of spiking, integrate and fire type of units and connections (on the order of 10^6). The connections are plastic with a Hebbian learning rule and the updates are event driven. Currently only a single chip implementation is discussed. In a similar line of work are NESPINN and its evolution MASPINN (Schæfer, Schnauer et al. 2002). The later is capable of running 10^6 spiking units and at the core of the computations is the NeuroPipe chip (Schoenauer, Atasoy et al. 2002). Systolic arrays are also typically implemented with ASIC:s. They have been used to both implement feed-forward networks trained with Back-propagation (Mahapatra and Mahapatra 2000) and Hopfield networks (Kung and Hwang 1988; Lehmann, Viredaz et al. 1993; Margaritis 1995).

REMAP was a Swedish project with the goal of building a massively parallel hardware for running neural networks, based on FPGA:s (Bengtsson, Linde et al. 1993). It was a standalone design and each FPGA implemented a number of bit-serial processors. Sparse distributed memory and self-organizing maps have been implemented on this machine. SPIKE128k is system based on a FPGA and developed for running spiking neural networks with Hebbian learning (Schæfer, Schnauer et al. 2002).

ParSPIKE is a development of SPIKE128k where the FPGA:s have been replaced with DSP:s (Schæfer, Schnauer et al. 2002). Another architecture based on an array of DSP:s is MUSIC (Muller, Gunzinger et al. 1995).

CNAPS was a general-purpose accelerator for neural networks that was fitted into regular desktop computers and it was developed by Adaptive Solutions (Hammerstrom 1995). Each chip had 64 arithmetic units and each arithmetic unit had access to a total of 4 KB of local memory. Up to eight chips, with a total of 512 arithmetic units, could be coupled to a host workstation. All chips were connected with two buses, which allowed for efficient broadcasts.

For an overview of FPGA designs see Girau (Girau 2000), and for an introduction to implementations of attractor networks on FPGA:s see Hammerstrom et al. (Hammerstrom, Gao et al. 2003).

8.2 Implementation of EWMA:s

Here we present a fixed-point arithmetic implementation of an EWMA as formulated in eq. (8.1) where $s \in (0,1)$ is a time varying input, p is a continuously updated estimate of s, and τ is a plasticity parameter that controls the speed by which the value of p approaches s. The fixed-point implementation presented here is proposed in appendix V. The EWMA, as formulated in eq. (8.1), can be interpreted as a leaky integrator. In appendix V we show that the EWMA can be computed in the logarithmic domain and that it then has a constant relative truncation error in the linear domain. This means that the proposed fixed-point arithmetic implementation can handle numerical values in a large range without serious round-off errors.

$$\tau \frac{dp(t)}{dt} = s(t) - p(t) \tag{8.1}$$

8.2.1 Conversion into the Logarithmic Domain

In the case of a floating-point arithmetic implementation it is straightforward to implement eq. (8.1) with Euler's method and the dynamical range of 32-bits floating-point variables is high enough to avoid serious round-off errors. But in the case of a fixed-point arithmetic implementation it is important that the dynamical range of the variables is used efficiently in order to minimize the required precision (number of bits). The weights of the BCPNN are computed based on the *p*-variables that often have values close to zero. Therefore it is important to have a high precision for small values and this can be achieved by computing with logarithmic values of the *p*-variables, here called the *W*-variables. Another reason for computing in the logarithmic domain is that the average number of operations per connection update is reduced when unary coded activity is used, which also a floating-point implementation benefits from. The number of operations needed to decay a connection are very few and if 1% of the units spikes each time step, most of the connections are decayed (99.99%) and only 0.01% of the connections are incremented (a computationally more expensive operation).

Here we show how the EWMA can be computed with a difference equation in the logarithmic domain, which is advantageous because the relative error is then constant (shown in appendix V). The starting point for the derivation is eq. (8.1) which is converted into eq. (8.2) by substituting *p* with a^W, and *dp* with $a^W \log a \, dW$:

$$\tau \frac{dW(t)}{dt} = \frac{1}{\log a}\left(s(t)a^{-W(t)} - 1\right) \tag{8.2}$$

Equation (8.2) cannot be effectively solved numerically with Euler's method because it is stiff. One solution is to use a more sophisticated numerical method but this means a large amount of complicated computations. Instead we analytically solve the differential equation in eq. (8.2), which gives us the difference equation (8.3) where the time-step is set to 1. The reason for the choice of the arbitrary base *a* in the calculations is that both the logarithm and exponential functions in eq. (8.4) and the constant decay factor in eq. (8.5) are easy to implement as integer valued look-up tables when the base for the computations can be chosen freely.

$$W_{t+1} = \log_a\left(s_t - e^{-1/\tau}(s_t - a^{W_t})\right) \tag{8.3}$$

An interesting property of eq. (8.3) is that if s=0 and a=e (the natural exponent) the equation is reduced to $W_{t+1}=W_t-1/\tau$. This means that W is subject to a linear decay when the input is zero, which requires very few operations. Having inputs equaling zero is typical for models of spiking neurons and hence our algorithm enables a very efficient implementation of these. Equation (8.3) is defined for $s\in(0,1)$ and $W\in[-\infty,0)$. But we intend to work with variables in the range $(0,M)$ and therefore we rewrite eq. (8.3) as eq. (8.4) where $s\in(0,M)$, $W\in(0,M)$.

$$W_{t+1} = M \log_M\left(s_t - e^{-1/\tau}(s_t - M^{W_t/M})\right) \tag{8.4}$$

Similarly, there is a linear decay when s=0:

$$W_{t+1} = W_t + M \log_M e^{-1/\tau} \tag{8.5}$$

The logarithmic values, W, of the EWMA can be converted back into the linear domain by:

$$p = M^{\frac{W}{M}-1} \tag{8.6}$$

8.2.2 Fixed-Point Arithmetic Implementation

In this section we discuss how the EWMA, transformed into the logarithmic domain, is implemented with discrete valued variables. The precision of these variables can range from 4 bits and up in the case of attractor network implementations (appendix V). In order to achieve a high precision on the average, despite the use of low precision variables, probabilistic fractional bit (PFB) (Hoehfeld and Fahlman 1992; Melton, Phan et al. 1992) is used. PFB means that repeated truncation errors in the computations are cancelled out over a period of time by the use of probabilistic calculations. Therefore can a fractional value be approximated accurately without bias as an average in time of integer values.

In the following we will assume that a discrete implementation is made with integer variables that have a precision of $\log_2(M+1)$ bits; $s\in(0,M)$ and $W\in(0,M)$. Random numbers are generated with a precision of $\log_2(K+1)$ bits in the range $(0,K-1)$ by the function $R()$. The value of the plasticity parameter, τ, is fixed but it is easy to extend the algorithm with multiple values of τ. To implement eq. (8.4) we use six look-up tables, three for the integer values; $T_1[x]=floor(M^{x/M})$,$T_2[x]=floor(xe^{-1/\tau})$, and $T_3[x]=floor(M\log_M x)$, and three for the fractional bits: $Tf_1[x]=floor((K+1)(M^{x/M}-T_1[x]))$, $Tf_2[x]=floor((K+1)(xe^{-1/\tau}-T_2[x]))$, and $Tf_3[x]=floor((K+1)(M\log_M x-T_3[x]))$. Here, $0\leq x\leq M$ and we set $T_3[0]=0$ and $Tf_3[0]=0$. The probabilistic computation is done in the $PFB(x,y)$-function; eq. (8.7). The constant decay, eq. (8.5), is implemented with PFB as well; $T_4=floor(-M\log_M e^{-1/\tau})$, $Tf_4=floor((K+1)(-M\log_M e^{-1/\tau}-T_4))$.

$$PFB(x,y) \leftarrow \begin{cases} T_y[x]+1 & if\ \left(R() < Tf_y[x]\right) \wedge \left(T_y[x] < M\right) \\ T_y[x] & otherwise \end{cases} \tag{8.7}$$

The computation of the W-variables is implemented with the look-up tables as in eq. (8.8), where we have defined $X\equiv s_t-PFB(W_t,1)$ and $Y\equiv PFB(0,4)$, where the latter term is the PFB-implementation of the constant decay.

$$W_{t+1} = \begin{cases} W_t - Y & if\ s_t = 0 \wedge W_t \geq Y \\ PFB\left(s_t + PFB\left(-X,2\right),3\right) & else\ if\ X < 0 \\ PFB\left(s_t - PFB\left(X,2\right),3\right) & otherwise \end{cases} \tag{8.8}$$

The computational requirement of the proposed implementation is 2 additions and 3 table lookups for an increment ($s>0$) and 1 addition and 1 table lookup for a decrement ($s=0$). In case of the BCPNN, the decrements usually outnumber the increments by a factor hundred or more. This means that on the average the computational requirement is close to that of the decrement. These figures should be compared with the naïve implementation of eq. (8.1) with Euler's method that requires 2 additions and 1 multiplication. Also, in the BCPNN algorithm the weights are computed directly from the logarithmic W-values so there is no need to convert them back into the linear domain. Further, the proposed implementation requires about 30% less memory for the W-variables than does a naïve implementation with Euler's method.

8.3 Fixed-Point Arithmetic Implementation of BCPNN

In section 8.2 we concluded that the computation of w_{ij} and β_i are best performed in the logarithmic domain. The potential, m_j, is not biased towards small or large values and therefore it is best implemented with Euler's method. In both cases PFB should be used to improve the average accuracy over time.

Here we consider the implementation of a P-type network with spiking units, i.e. AER is used. In this case we can compute the weights as:

$$w_{ij} = \log\left(\frac{p_{ij}}{p_i p_j}\right) = \log(p_{ij}) - \log(p_i) - \log(p_j) \tag{8.9}$$

We now make the following definitions; $W_{ij} \equiv \log(p_{ij})$, $W_i \equiv \log(p_i)$, $W_j \equiv \log(p_j)$. The weights are computed in eq. (8.10) and (8.11) where >> is the right-bit-shift operator.

$$\beta_i = \begin{cases} 0 & \textit{if } W_i = 0 \\ (M + W_i) >> 1 & \textit{otherwise} \end{cases} \tag{8.10}$$

$$w_{ij} = \begin{cases} M >> 1 & \textit{if } W_i = 0 \vee W_j = 0 \\ 0 & \textit{else if } W_{ij} = 0 \\ (2M + W_{ij} - W_i - W_j) >> 1 & \textit{otherwise} \end{cases} \tag{8.11}$$

All β_i, w_{ij}, and W_{ij} have $\log_2(M+1)$ bits precision. The potential is computed as:

$$m_j(t+1) = m_j(t) + \left(T_5\left(\begin{matrix} s_j(t) - \\ \max\limits_{k \in X_h}(s_k(t)) \end{matrix} \right) - m_j(t) \right) \Big/ \tau_{\mathrm{m}} \quad \text{for each } h = \{1,2,...,H\} \tag{8.12}$$

The support, s_j, is a sum of at most H weights and has the precision $\log_2((M+1)H)$ bits. The support value is then truncated before it is used in the computation of the potential, m_j, which has $\log_2(M+1)$ bits. The look-up table for the exponential is computed as $T_5[x] = \mathit{floor}(Me^{(x-M)G/M})$, $Tf_5[x] = \mathit{floor}((K+1)(Me^{(x-M)G/M} - T_5[x]))$ where $1 \le x \le M$, and $T_5[0]=0$, $Tf_5[0]=0$.

For the fixed-point arithmetic implementation the scaling of the memory is the same as for a network implemented with floating-point arithmetic but with a smaller factor. Here, a connection in a P-type network is implemented with 7 bytes, which is 30% less than for the floating-point implementation. First, 8 bits are used for the W_{ij}, 32 bits (21 bits is sufficient to give all units in a H-sized BCPNN a unique identifier, but there is a large overhead using custom sized integers on a general purpose processor) for the index of the presynaptic unit, and 2 bytes for the control of the event driven updating.

The reduced memory requirement of the connection variables has a positive spin-off effect in that a smaller communication bandwidth between the processor and memory is needed. In the case of implementations on parallel computers, the fixed-point arithmetic version is actually slightly faster than the floating-point one (section 7.5.3).

A disadvantage with using PFB is that random numbers are needed, but if PFB is not used each of the W-variables needs to be stored with about 20 bits instead of 8 bits (Johansson and Lansner 2004b; Johansson and Lansner 2004a). This applies both to our and the naïve implementation of the EWMA.

8.4 Implementation of BCPNN on the Mitrion Co-Processor

The Mitrion Co-Processor is a board that is fitted into a regular desktop computer. The board is equipped with a single FPGA and a few megabytes of SRAM memory. To use the board, computational intensive sections of a program must be rewritten in a C like language developed by the Mitrion company. The FPGA is configured using this code to perform the calculations of the computational intensive parts of the program. The key to performance is to effectively use the massive parallelism available in the FPGA, which is non-trivial for most applications.

A quick description of the execution pattern of a program running on a computer equipped with a Mitrion board is now given: The program on the host computer executes up to the point of a piece of code designed to run on the FPGA. Data is then downloaded from the program to the Mitrion board and the FPGA. Next the computations are performed in parallel on the FPGA. Finally, the results from these computations are uploaded back to the program running on the host computer.

In order to achieve good performance it is important that the program execution remain in the computational intensive parts of the code for long periods of time, because the communication between the FPGA and the host computer's processor is slow. It is also important, for performance reasons, that the downloaded code uses the off-chip SRAM memory on the board as little as possible because the access to this memory is a bottleneck. Unfortunately, the memory on-chip with high-speed access is limited to a few hundred Kbytes in today's FPGA:s. This is a problem for memory intensive applications such as large-scale neural networks that have to store a massive number of connection weights. The largest network we have implemented had its entire weight matrix stored in the fast on-chip memory on the FPGA.

We have implemented a BCPNN of the counting type with 400 units on a Mitrion Co-Processor board (Andersson, Johansson et al. 2006). The preliminary results show that we get almost one order of magnitude speed-up compared with an implementation on a standard Pentium 4 desktop computer. This result is promising but in order to implement large-scale networks with millions of units, high-speed access to large amounts of memory must be made available to the FPGA. Dan Hammerstrom has been working on such a design but due to shortages in funding its development is for the moment put on hold.

A similar product as to Mitrion's where an FPGA is programmed using a high level C like language is offered by the company Celoxica. Their product is a board with a FPGA and a few megabytes of external SRAM bundled together with a development environment. The FPGA is programmed using a language named Handel-C. We have done some preliminary studies on using their product with promising results.

9 Discussion

In this thesis we have proposed a generic model of the mammalian neocortex and discussed its properties and implementation. The objective of this work was twofold: firstly, we wanted to build a platform that can be used to peek into the workings of cortex; secondly, we wished to create a framework from which new efficient solutions for problems in computer science can spring. In the latter case it is important to realize that these brain-inspired algorithms are not able to compete with state-of-the-art algorithms when running on a single processor, but require massively parallel hardware to be competitive.

In this chapter we discuss aspects and issues of the model, its characteristics, and its implementation. In the last section of the chapter we discuss the road a head for the computers of tomorrow.

9.1 The Cortical Model

In the process of designing the abstract cortical model we reviewed a sizeable number of papers on the subjects of size and structure of the mammalian cortex. When studying these data one is stroke by the large differences that exists between individuals, e.g. the size of V1 can differ by a factor 2-3 between different individuals. As a result, variations also exist between different studies of the same cortical area in the same species. The existence of these large size variations of cortical areas indicates that cortex is a very flexible computing system. In the beginning of this thesis we quantified the cortices of five different mammals in terms of neuron and synapse numbers, spatially arrangement, and connectivity patterns. We found the same general characteristic structures and organization principles in all five species; the modularization of cortex into columns and layers. Both hypercolumns and minicolumns were found to have similar dimensions and structure in different cortical areas and species. All this data was considered when the abstract model of cortex was formulated.

When designing a functional model of a system it is important to wisely choose the level of abstraction, especially when the system has a huge number of free variables such as the neocortex. We chose the level of abstraction mainly on the basis of anatomical and electrophysiological data. The functional unit in our model, the cortical minicolumn, was modeled by a small number of coupled first order ordinary differential equations. The minicolumn can of course be modeled by a more complex set of equations e.g. as a group of integrate and fire neurons (Lücke 2004; Lücke and Malsburg 2004), but we have not come across any facts that have led us to reconsider our current model given the focus on computational functionality. The proposed model complements more biologically detailed compartmental models such as that of Lundqvist et al. (Lundqvist, Rehn et al. 2006b).

A reasonable question is why bunt a hundred neurons into a single minicolumn unit that can be described with a limited number of parameters (in our model up to three) instead of using the full representational strength of these neurons which hypothetically could amount to a couple of hundred or so variables. We advocate this approach by the following three arguments: I) It gives redundancy; a minicolumn is not affected if a few neurons die. II) It gives high connectivity, which is necessary for an appropriate functionality. III) It gives the ability to use a real-valued coding of activity.

The functionality of the hypercolumn as a normalizing structure is based on experimental data (section 2.7.3), which has showed that an area in cortex corresponding to a hypercolumn has neurons arranged into minicolumns that are selectively responsive for different line orientations. In our model we implement the hypercolumn as module of minicolumn units. The firing frequency of these units is controlled by a soft winner-take-all function taken over the units' synaptic input, within the hypercolumn. We showed that the main advantage of using this type of coding in case of a recurrently connected network is that it gives a stable activity level.

In our view of neocortex, its computational capabilities are mainly decided by the number of synapses it possesses. It has therefore been our goal to create a model that incorporates the same total number of synapses as cortex. Usually, in cortical models, the synapses are sub-sampled so that only a small fraction of all are modeled and instead there is a focus to simulate all neurons, which are 10^4 fewer than the synapses.

In the current implementation of the cortical model the long-range recurrent connections are set up randomly. There is strong evidence that this is not the case in the mammalian cortex (Young, Scannell et al. 1994; Scannell, Blakemore et al. 1995; Sporns, Tononi et al. 2000). Neither did we consider conduction delays in the cortical signaling. The mean axon diameter is 26% larger in humans than in mice, while a 1500% increase would be needed to maintain the same conduction delays (Kaas 2000). Obviously, for dynamical systems such as cortex, conduction delays can have a large influence on their behavior and this is something that must be investigated in a future study. Finally, reducing the randomness of the connectivity and introducing conduction delays into the model will help to make implementations of it more efficient (Djurfeldt, Johansson et al. 2005).

The current cortical model does not include any interactions between cortex and other parts of the brain such as hippocampus, amygdala, basal ganglia, cerebellum, or thalamus. Obviously, these components fill important roles in a brain-like processing system e.g. hippocampus is very important for memory functions, amygdala for the processing of feelings and emotions, and basal ganglia for motor commands (Rolls and Treves 1998; Balkenius and Morén 2001). In the case of vision a lot of preprocessing is taking place already in LGN in thalamus and also in the retina, and these parts must undoubtedly be accounted for in a model of the human visual system.

The current model lacks an internal ordering of the stored memories and it does not consider the aspect of time. In its current form the model is indifferent to the timescale that it operates on. In a future, more realistic model of the brain and cortex, the concept of time must be incorporated. Extending the cortical model along these lines with models of more parts of the brain and incorporating the notion of time are clearly very important long-term goals.

9.2 Computational Aspects of the Model

Hypercolumnar attractor networks are very similar to more traditional attractor networks where the units have individually set thresholds. One of the big strength of columnar networks is their robustness. The normalization performed by the hypercolumns keep the activity level constant, which is important for having good attractor dynamics. Also, because all inputs are normalized there is less of a problem with varying levels of fan-in, i.e. different levels of synaptic input due to different numbers of incoming connections and noisy input. Hypercolumnar structured networks have less of a problem with spurious attractors. Finally, we conclude that a partially diluted and asymmetric weight-matrix does not lead to a catastrophic degradation of storage capacity although convergence cannot be guaranteed in this case. This is an important conclusion since a biological attractor network most certainly not has a connection matrix that is perfectly symmetric.

In order to get an attractor network with high storage capacity it is necessary to use sparse coding and this is particular important when using a network with weights that have limited precision e.g. a Willshaw-Palm network. In the cortical model the hypercolumns have a fixed size and hence in a naively implemented network there is a fixed activity level. Increasing the size of such a network by adding more hypercolumns results in poor scaling of the number of stored memories. We showed that by using “quiescent hypercolumns” the coding level in the network could be chosen independently of the hypercolumn size, which enabled good scaling of the storage capacity.

The cortical model when formulated as a large attractor network tends to have the properties of a small-world network, i.e. a short average path length between any two units and a high clustering, due to its patchy connectivity. Further, we found that the structured patchy connectivity increased the storage capacity by improving the signal-to-noise ratio of the synaptic input.

We showed that learnt memories could be instantaneously clustered to different degrees by regulating a single parameter, G. In the extreme case of low G-values only a single memory (attractor) remained in the network. This feature, being able to choose an arbitrary level of memory clustering, is not found in more traditional clustering techniques where the number of clusters must be decided on before hand. The usefulness of this feature in a memory system is obvious; for example we can chose if we want to consider a particular car or the more general concept of a car.

Memory clustering is an example of an operation that is intrinsically performed by attractor networks but hard to implement with a normal computer program. We believe that identifying and studying this type of intrinsic computational capabilities of attractor networks is important if we want to understand the brain and build brain-like-computers.

In connection with the experiments on clustering we derived the energy function for a BCPNN with spiking activity and also the energy function for an approximation of BCPNN with real-valued activity. Empirically, the BCPNN has been found to always converge to stable fix-points when the matrix is symmetric and this suggests that there could be an energy function for this network (Sandberg 2003).

We showed that a section taken from the hierarchical cortical model could be used as an auto-associative memory in which patterns not possible to store in a single layered attractor network could be stored. Constructing networks of networks is an interesting topic of neural network research that so far has received little interest, mainly because of the high complexity associated with these designs. Although, we foresee a steady progress towards evermore complex systems based on these types of designs (Deco and Schürmann 2000; Deco 2001; Deco and Lee 2002; Deco and Rolls 2004).

9.3 Implementation Aspects of the Model

We discussed different levels of parallelism in neural networks and showed that an implementation of the cortical model using parallelism over hypercolumns scaled well on both traditional cluster computers such as Lenngren and on massively parallel clusters such as Blue Gene/L. As we discussed in section 9.1 the scaling would be even better if the connectivity was ordered instead of random and if also asynchronous communication was used.

We showed that it was possible to run up to rat-cortex sized equivalents of the cortical model and we extrapolated the computational requirements for running human sized networks. The results from these simulations showed that the networks were not constrained by the communication at their present size. When the networks increase in size, communication risk to become a bottleneck, but by moving away from completely random connectivity we think that this will not become a problem.

We developed a fixed-point arithmetic implementation of the BCPNN algorithm and it was shown to reduce the memory requirements by 30% compared to a floating-point arithmetic implementation when running on cluster computers. This implementation makes it possible to implement BCPNN in digital hardware, e.g. FPGA:s.

On a cluster connected with a standard 100 Mbits Ethernet we showed that global communication could be efficiently implemented using UDP/IP based communication. The drawback with UDP/IP based communication for many applications is that there can be errors in the communication, such as lost and late arriving packages. However, all iterative algorithms that similar to attractor networks can absorb small amounts of errors can gain in speed by using this type of communication.

An interesting aspect of large-scale abstract implementations of attractor networks is that the methodological difficulties faced are very similar to those faced when creating more biologically realistic models of the brain with the Hodgkin-Huxley formalism. The solutions to inter-unit communication, memory access patterns, and algorithmic implementation are all similar and, hence, technical solutions to these problems can be used in both the abstract and the more biologically detailed simulations. The abstract models have an advantage over the more biologically detailed ones in that full-scale models can be run. The more detailed models can in a direct way be coupled to a neuron's underlying chemistry with signal substances and neuromodulators, but this is also possible for our abstract model as demonstrated in appendices III and VIII.

9.4 Feature Extraction in the Model

One aspect that needs to be developed in the proposed hierarchical model is the feature extraction that takes place in the projection from a lower to a higher area. As for now we use simple competitive learning, but we expect that a more refined algorithm can enhance the capabilities of the system. We want the feature algorithm to be unsupervised, possibly with the option of incorporating a reinforcement signal. We would also like to have an algorithm that provides size, position, and other forms of invariances. Further, it is possible that we actually want to have different types of feature extraction algorithms at different levels in the hierarchical system. Becker (1996) has proposed that the goal of the first levels in the cortical processing hierarchy is to reduce dimensionality while maximizing the information transmitted. At higher levels, factors such as prior knowledge about the structure in the world may play a part determining how the nervous system filters information and shapes our perception (Becker and Plumbley 1996). In the following we take a brief overview of the different techniques that can be used for feature extraction.

Becker and Plumbley (1996) identify three categories of unsupervised methods that can extract features (Becker and Plumbley 1996); information-preserving methods, density estimation methods, and feature extraction methods.

The major algorithms in the first class of information-preserving methods are principal component analysis (PCA) and independent component analysis (ICA) (Haykin 1999). There is also the *Infomax* principle by Linsker, which states that the mutual information should be maximized between the input and the output of the feature extraction network (Linsker 1988). Further, Barlow has proposed that the redundancy of an invertible code should be minimized, i.e. the hidden units' outputs should be statistically independent when conditioned on the input (Barlow 1989). In the general case, this problem is intractable. But when the solution is approximated it gives the independent components that build up a factorial code of the input (Schmidhuber 1992). In fact, also Linsker's Infomax principle can be used to derive the independent components of an input (Bell and Sejnowski 1997; Obradovic and Deco 1998). Interestingly, these components are also possible to compute by assuming a sparse coding in the hidden layer (Olshausen and Field 1996; Olshausen and Field 1997). Both PCA (Rubner and Schulten 1990) and ICA (Jutten and Herault 1991; Karhunen, Oja et al. 1997) can be implemented with neural networks that have competitive learning in a feedforward projection from the input to the hidden layer, and a hidden layer with lateral inhibition. There are a large number of networks with a similar structure that operates in a similar way, creating decorrelated representations of the input data (Földiak 1990; Schraudolph and Sejnowski 1992; Dayan and Zemel 1995; Marshall 1995; Saund 1995; Harpur and Prager 1996; Deco and Parra 1997; Charles and Fyfe 1998; Hochreiter and Schmidhuber 1999; Spratling and Johnson 2002; Lücke 2004; Lücke and Malsburg 2004; Rao, Cecchi et al. 2005). A problem with many of these feature extraction networks is that they do not produce sparse distributed representations, which is necessary for good performance when using Hebbian types of learning-rules (Field 1994).

In case of density estimation methods, competitive learning (CL) (Rumelhart and Zipser 1985; Grossberg 1987), expectation maximization (EM), and self-organizing maps (SOM) are the most well known algorithms. A large number of different variants of CL have been proposed. They differ mainly in how the problem with dead units is solved, i.e. how unused code vectors are avoided. We use competitive selective learning, CSL, which theoretically offers no dead units and a minimal variance in the representation (Ueda and Nakano 1994).

The third approach is based on the idea to learn particular features of the input data rather then being able to fully account for all of this data, i.e. being able to reconstruct it. Here there are a number of different algorithms that create representations that either maximize or minimize the mutual information between created features. This type of algorithms uses spatio-temporal and multi-sensory coherence to give cues for segmenting signals in space and time, and for object localization and identification (Becker and Hinton 1992; Yuille, Smirnakis et al. 1995; Becker 1996; Back and Trappenberg 2001). In case of supervised learning there are also efficient feature extraction algorithms that greatly improve the results over using the simple learning vector quantization (LVQ) and unsupervised techniques such as PCA (Battiti 1994; Torkkola 2003).

In case of the naïve Bayesian classifier it is well known that the discretization method used plays a crucial role for achieving a good classification performance (Yang and Webb 2002). The situation is similar for the hierarchical system when used as a classifier, the choice of feature extractor from the input to the hidden population is very important (Steinert, Rehn et al. 2006).

For our cortical model it also important to have a feature extraction algorithm that generates a coding that fits the hypercolumnar structure and one that can be used in an attractor network, i.e. a unary coding. This is exactly the type of coding generated by CSL.

PCA and ICA generate real-valued factorial codes in the hidden population, which do not fit into our framework of hypercolumns and attractor dynamics. But there are algorithms that are designed to extract independent components in a way such that a sparse binary code is created (Rehn and Sommer 2006). Adapting that kind of algorithms to fit into the hypercolumn framework is an interesting topic for future research. An other interesting alternative to the PCA and ICA algorithms are the ones that combine both CL and information theoretic measures (Kamimura 2005), i.e. categories two and three in Becker and Plumbley's nomenclature.

One way of creating invariant recognition is to combine feature detectors in a hierarchy as in the Neocognitron (Fukushima 1988; Fukushima 1989; Fukushima and Wake 1991) or as in VisNET (Stringer and Rolls 2000; Elliffe, Rolls et al. 2002) and derivations of VisNET (Deco and Rolls 2004). In case of the Neocognitron the spatial invariance is created by simply duplicating the learnt feature detectors to cover the entire input view. But by assuming temporal constancy, several types of transformation and translation invariances, including spatial invariance, can be created in a biologically plausible way (Földiák 1991; Oram and Földiák 1996; Parga and Rolls 1998; Fukushima 1999; Stringer, Perry et al. 2006).

9.5 Notes on the Future of Computers

The classical processor architecture is the von Neumann statemachine that is controlled by a sequence of instructions. This design was invented in the late 1940's and the basic principle of this design is still used in today's processors. By increasing the rate by which processors decode instructions their speed has been increased orders of magnitude during the last 60 years. This has been achieved by increasing the clock frequency, but unfortunately this has become increasingly problematic because of problems with heat dissipation. The heat generated increases quadratically with the clock frequency. This has lead to a strive for increasing the parallelism in processors, and today's processors are typically capable of performing 3-6 instructions simultaneously. By using massively parallel designs, high clock frequencies are less important and this in turn also solves the problem with heat dissipation. For this type of designs the transistors are used more efficiently (assuming they have been programmed properly) than in the von Neumann machines because fewer transistors need to be assigned to caches and scheduling logic, and instead they can be used to implement arithmetic units.

Neural networks have been proposed to have a key role in future computer science. Maass advocates this by three arguments (Maass 2001): Firstly, the speed-up of today's processors has almost hit the physical barrier, clock rates cannot be turned up much more. Parallelism is the key to speed-up in future processors. Secondly, the computational applications of tomorrow will be more focused on interacting with humans and the world in a natural way. Hammerstrom refers to these computational tasks as boundary problems (Hammerstrom 2000). Tasks such as speech and visual recognition can gain a lot from parallel processing and they can often be effectively solved with neural networks already today. Thirdly, today's processors are very energy consuming and adopting a connectionist approach with neural networks where massively parallelism is used allows for power efficient designs.

Hammerstrom (2000) has suggested that exploiting the algorithmic robustness of neural networks is the key to efficient use of future hardware. Parallel computers and digital hardware that is used today are not ideal for running neural networks that do not really require deterministic and error free computations and reliable communication. We have shown that by removing some of the control mechanisms and accepting errors in the communication, substantial gains in speed and scalability can be achieved on some types of hardware (section 7.6.2). This issue will become even more important in the future when transistors get smaller and the noise levels in hardware increase due to quantum effects such as electron tunneling between wires on the chip. The cost of sheltering applications from increased noise levels is going to raise drastically, particular in future nanotechnology devices. Providing noise and error insensitive, high performance, algorithms for these molecular sized structures will be essential (Hammerstrom 2000). A conclusion made by Hammerstrom is that we are approaching an inflection point where neural network algorithms and hardware have been developed enough for it to be feasible to implement cortex-sized systems. This in turn will create a multitude of new applications and new solutions to old problems in computer science.

Already today are molecular scale technologies starting to emerge and a promising new technology is CMOL (Likharev, Mayr et al. 2003). In a CMOL chip nanowires are combined with traditional CMOS technology, which achieves an extremely high density of logic gates; a 100 cm^2 silicon wafer could potentially hold 10^{10} units with 10^4 connections each (Likharev, Mayr et al. 2003). Using this technology both recurrent and feed-forward neural networks can be implemented (Türel, Lee et al. 2005).

10 Conclusions and Future Work

In this thesis we have studied the mammalian cortex to come up with an abstract model of it. We have shown that this model can be used to investigate cortical anatomy and physiology, and it can also be seen as forming the basis for a new breed of massively parallel brain-inspired algorithms. We expect that further developments of this model, along the line as a model of the mammalian cortex, will both bring it an increased explanatory power but also enhance its computational capabilities. We have shown that the basic framework of this model is very flexible and robust, and that it scales well in terms of capability e.g. storage capacity. Because of its modular and inherently parallel structure efficient parallel implementations of it are possible, as we have shown, and these implementations do also scale well and do not have communication bottlenecks.

As the number of processors in today's computers is steadily increasing we expect to see brain-inspired, inherently parallel, algorithms to become more commonly used. In the year 2004 it was uncommon to have more than two processors in a desktop computer, but today, in the year 2006, it is possible to buy a desktop computer with up to 16 processors. This development towards parallel computers has just begun and we expect this trend to accelerate in the coming years, and when molecular scale computers arrive massive parallelism will be an integral part of their function. Brain-inspired algorithms are well positioned to take full advantage of these coming developments in computer technology.

One of the first areas of application for brain-inspired algorithms will probably be complex control systems that receive inputs from a large number of different sensors. The design of such control systems is very hard with today's techniques. Ensuring good quality, i.e. error free operation, of these systems is also difficult. Here, brain-inspired algorithms that learn from examples, adapts to the particular characteristics of the different sensors, are self-organizing, and self-repairing to some extent will improve both the performance and quality of complex control systems.

We envision that the development of both large-scale abstract and biologically detailed models of the neocortex, and also the brain, will accelerate in the coming years. These future models will be crucial in the development of treatments for psychiatric and neurological disorders, to guide experimentalists to interesting topics of study, and to track the emergence of intelligence and complete the puzzle of understanding of how the brain works.

The most important contribution this thesis makes to the understanding of the neocortex is that it gives a computational account for why synaptic connections are arranged into patches. In this thesis we also demonstrate that the modular structure of neocortex gives many computational advantages such as scalability and robustness. It is also predicted in this thesis that only a small fraction of the hypercolumn modules should be in an active state at any one time in the neocortex.

We end this thesis by leaving some pointers to interesting topics for future studies. The work in this thesis followed up on the directions proposed in an earlier licentiate thesis by the same author (Johansson 2004).

There are a number of things about the BCPNN model that should be investigated in future studies: Firstly, finding an energy function for the exact formulation of the BCPNN algorithm. Secondly, continuing the work on hardware implementations, primarily using

FPGA:s. Thirdly, deducing more information on the principles underlying the hierarchical clustering of memory patterns in attractor networks. Fourthly, the connection to hierarchical Bayesian inference and graph models should be investigated.

But the most interesting topic for a future study, as I see it, is to continue the development of the hierarchical cortex model. As mentioned in the preceding section the algorithms used for feature extraction can be refined and doing so should improve e.g. the classification performance on the MNIST data. Also, incorporating different forms of invariances into the system and making it able to incorporate reinforcement-based learning is also important. With these extensions in place the system may be starting to have the functional capabilities necessary for it to compete with state-of-the-art algorithms in different areas of application. In a long-term perspective of thirty to forty years we envision to see implementations of this type of brain-inspired computational structures in molecular scale technologies where they are made no larger than a brain and also with equivalent power consumption. In this thesis we took the first steps towards these goals.

References

Abbott, L. F. (1990). Learning in neural network memories. *Network: Comp. in Neur. Sys.* 1(1): 105-122.

Abbott, L. F. and Y. Arian (1987). Storage capacity of generalized networks. *Physical Review A* 36(10): 5091-5094.

Abeles, M., E. Vaadia, et al. (1990). Firing patterns of single units in the prefrontal cortex and neural network models. *Network: Comp. in Neur. Sys.* 1(1): 13-25.

Abraham, W. C. and M. F. Bear (1996). Metaplasticity: the plasticity of synaptic plasticity. *Trends in Neurosci.* 19(4): 126-130.

Alonso, J.-M. (2002). Neural connections and Receptive Field Properties in the Primary Visual Cortex. *The Neuroscientist* 8(5): 443-456.

Alonso, J.-M. and L. M. Martinez (1998). Functional connectivity between simple cells and complex cells in cat striate cortex. *Nature Neurosci.* 1(5): 395-403.

Amari, S. I. (1977). Neural theory of association and concept-formation. *Biol. Cybern.* 26: 175-185.

Amirikian, B. and A. P. Georgopoulos (2003). Modular organization of directionally tuned cells in the motor cortex: Is there a short-range order? *Proc. Natl. Acad. Sci. USA* 100(21): 12474-12479.

Amit, D. J. and S. Fusi (1992). Constraints on learning in dynamic synapses. *Network: Comp. in Neur. Sys.* 3(4): 443-464.

Amit, D. J. and S. Fusi (1994). Learning in Neural Networks with Material Synapses. *Neural Computation* 6(5): 957-982.

Amit, D. J., S. Fusi, et al. (1997). Paradigmatic Working Memory (Attractor) Cell in IT Cortex. *Neural Computation* 9(5): 1071-1092.

Amit, D. J., H. Gutfreund, et al. (1985). Spin-glass models of neural networks. *Physical Review A* 32(2): 1007-1018.

Amit, D. J., H. Gutfreund, et al. (1987). Information storage in neural networks with low levels of activity. *Physical Review A* 35(5): 2293-2303.

Amit, Y. and M. Mascaro (2001). Attractor Networks for Shape Recognition. *Neural Computation* 13(6): 1415-1442.

Andersson, U., C. Johansson, et al. (2006). *A Mitrion-C implementation of a Bayesian Confidence Propagating Neural Network*, Royal Institute of Technology and Uppsala University, Stockholm.

Andrews, T. J., S. D. Halpern, et al. (1997). Correlated Size Variations in Human Visual Cortex, Lateral Geniculate Nucleus, and Optic Tract. *J. Neurosci.* 17(8): 2859-2868.

Athithan, G. and C. Dasgupta (1997). On the problem of spurious patterns in neural associative memory models. *IEEE Trans. on Neural Networks* 8(6): 1483-1491.

Attwell, D. and S. B. Laughlin (2001). An Energy Budget for Signaling in the Grey Matter of the Brain. *J. Cereb. Blo. Flo. Met.* 21(10): 1133-1145.

Back, A. D. and T. P. Trappenberg (2001). Selecting inputs for modeling using normalized higher order statistics and independent component analysis. *IEEE Trans. on Neural Networks* 12(3): 612-617.

Badoni, D., S. Bertazzoni, et al. (1995). Electronic implementation of an analogue attractor neural network with stochastic learning. *Network: Comp. in Neur. Sys.* 6(2): 125-157.

Bailey, J. and D. Hammerstrom (1988). *Why VLSI Implementations of Associative VLCNs Require Connection Multiplexing*: 173-180. International Conference on Neural Networks (ICNN'88), San Diego, IEEE.

Bair, W. and C. Koch (1996). Temporal Precision of Spike Trains in Extrastriate Cortex of the Behaving Macaque Monkey. *Neural Computation* 8: 1185-1202.

Balkenius, C. and J. Morén (2001). Emotional Learning: A Computational Model of the Amygdala. *Cybernetics and Systems* 32: 611-636.

Bar, M. (2003). A Cortical Mechanism for Triggering Top-Down Facilitation in Visual Object Recognition. *J. Cog. Neurosci.* 15(4): 600-609.

Barabási, A.-L. and R. Albert (1999). Emergence of Scaling in Random Networks. *Science* 286(5439): 509-512.

Barlow, H. B. (1989). Unsupervised Learning. *Neural Computation* 1(3): 295-311.

Bartfeld, E. and A. Grinvald (1992). Relationships between orientation-preference pinwheels, cytochrome oxidase blobs, and ocular-dominance columns in primate striate cortex. *Proc. Natl. Acad. Sci. USA* 89: 11905-11909.

Bartlett, M. S. and T. J. Sejnowski (1998). Learning viewpoint-invariant face representations from visual experience in an attractor network. *Network: Comp. in Neur. Sys.* 9(3): 399-417.

Barton, R. A. (2002). How did brains evolve? *Nature* 415: 134-135.

Barton, R. A. and P. H. Harvey (2000). Mosaic evolution of brain structure in mammals. *Nature* 405(6790): 1055-1058.

Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *IEEE Trans. on Neural Networks* 5(4): 537-550.

Beaulieu, C. (1993). Numerical data on neocortical neurons in adult rat, with special reference to the GABA population. *Brain Res.* 609: 284-292.

Beaulieu, C. and M. Colonnier (1989). Number and Size of Neurons and Synapses in the Motor Cortex of Cats Raised in Different Environmental Complexities. *J. Comp. Neurol.* 289: 178-187.

Beaulieu, C., Z. Kisvarday, et al. (1992). Quantitative Distribution of GABA-immunopositive and -immunonegative Neurons and Synapses in the Monkey Striate Cortex (Area 17). *Cerebral Cortex* 2(4): 295-309.

Becker, S. (1996). Mutual information maximization: models of cortical self-organization. *Network: Comp. in Neur. Sys.* 7: 7-31.

Becker, S. and G. E. Hinton (1992). Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature* 355: 161-163.

Becker, S. and M. Plumbley (1996). Unsupervised Neural Network Learning Procedures For Feature Extraction and Classification. *J. Applied Intelligence* 6: 185-205.

Bell, A. J. and T. J. Sejnowski (1997). The Independent Components of Natural Scenes are Edge Filters. *Vision Research* 37(23): 3327-3338.

Bender, K. J., J. Rangel, et al. (2003). Development of Columnar Topography in the Excitatory Layer 4 to Layer 2/3 Projection in Rat Barrel Cortex. *J. Neurosci.* 23(25): 8759-8770.

Bengtsson, L., A. Linde, et al. (1993). *The REMAP massively parallel computer platform for neural computations*: 47-62. International Conference on Microelectronics for Neural Networks, Edinburgh, Scotland, UK.

Bengtsson, M. and J. Schubert (2001). Dempster-Shafer clustering using Potts spin mean field theory. *Soft Computing* 5(3): 215-228.

Bi, G.-q. and M.-m. Poo (2001). SYNAPTIC MODIFICATION BY CORRELATED ACTIVITY: Hebb's Postulate Revisited. *Annu. Rev. Neurosci.* 24: 139-166.

Bi, G. and M. Poo (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci.* 18(24): 10464-10472.

Blasdel, G. G. (1992a). Differential imaging of ocular dominance and orientation selectivity in monkey striate cortex. *J. Neurosci.* 12(8): 3115-3138.

Blasdel, G. G. (1992b). Orientation selectivity, preference, and continuity in monkey striate cortex. *J. Neurosci.* 12(8): 3139-3161.

Bode, B., J. Hill, et al. (2004). *Cluster Interconnect Overview*: 211-218. USENIX Extreme Linux Technical Conference, Boston, Massachusetts, July 2004.

Bohland, J. W. and A. A. Minai (2001). Efficient associative memory using small-world architecture. *Neurocomputing* 38-40: 489-496.

Bollé, D., P. Dupont, et al. (1992a). On the phase diagram of the Q-state Potts-glass neural network. *Physica A* 185: 363-368.

Bollé, D., P. Dupont, et al. (1992b). Thermodynamic properties of the Q-state Potts-glass neural network. *Phys. Rev. E* 45(6): 4194-4197.

Bollé, D., P. Dupont, et al. (1991). Stability properties of Potts neural networks with biased patterns and low loading. *Journal of Physics A* 24(5): 1065-1081.

Bollé, D., P. Dupont, et al. (1992c). On the overlap dynamics of multi-state neural networks with a finite number of patterns. *J. Phys. A: Math. Gen.* 25(10): 2859-2872.

Bollé, D. and F. Mallezie (1989). Image evolution in Potts-glass neural networks. *Journal of Physics A* 22(20): 4409-4418.

Bonnaz, D. (1997). Storage Capacity of Generalized Palimpsests. *J. Physique* 7: 1709-1721.

Bosch, H. and F. J. Kurfess (1998). Information storage capacity of incompletely connected associative memories. *Neural Networks* 11(5): 869-876.

Bosking, W. H., Y. Zhang, et al. (1997). Orientation Selectivity and the Arrangement of Horizontal Connections in Tree Shrew Striate Cortex. *J. Neurosci.* 17(6): 2112-2127.

Bourgeois, J. P. and P. Rakic (1993). Changes of synaptic density in the primary visual cortex of the macaque monkey from fetal to adult stage. *J. Neurosci.* 13(7): 2801-2820.

Boyd, J. and J. Matsubara (1991). Intrinsic connections in cat visual cortex: a combined anterograde and retrograde tracing study. *Brain Research* 560(1-2): 207-215.

Braitenberg, V. (2001). Brain Size and Number of Neurons: An Exercise in Synthetic Neuroanatomy. *Journal of Computational Neuroscience* 10(1): 71-77.

Braitenberg, V. and A. Schuz (1998). *Cortex: Statistics and Geometry of Neuronal Connectivity*. New York, Springer Verlag.

Brunel, N., F. Carusi, et al. (1998). Slow stochastic Hebbian learning of classes of stimuli in a recurrent neural network. *Network: Comp. in Neur. Sys.* 9(1): 123-152.

Brunso-Bechtold, J. K., M. C. Linville, et al. (2000). Age-related synaptic changes in sensorimotor cortex of the Brown Norway X Fischer 344 rat. *Brain Res.* 872(1-2): 125-133.

Buckingham, J. and D. Willshaw (1992). Performance characteristics of the associative net. *Network: Comp. in Neur. Sys.* 3(4): 407-414.

Buckingham, J. and D. Willshaw (1993). On setting unit thresholds in an incompletely connected associative net. *Network: Comp. in Neur. Sys.* 4(4): 441-459.

Buhmann, J., R. Divko, et al. (1989). Associative memory with high information content. *Physical Review A* 39(5): 2689-2692.

Buldyrev, S. V., L. Cruz, et al. (2000). Description of Microcolumnar Ensembles in Association Cortex and their Disruption in Alzheimer and Lewy Body Dementias. *Proc. Natl. Acad. Sci. USA* 97(10): 5039-5043.

Buxhoeveden, D., A. Switala, et al. (2000). Quantitative analysis of cell columns in the cerebral cortex. *J. Neurosci. Meth.* 97: 7-17.

Buxhoeveden, D. P. and M. F. Casanova (2002). The minicolumn hypothesis in neuroscience. *Brain* 125(5): 935-951.

Buxhoeveden, D. P., A. E. Switala, et al. (2001). Morphological Differences Between Minicolumns in human and Nonhuman Primate Cortex. *Am. J. Phys. Anthropol.* 115(4): 361-371.

Callaway, E. M. (1998). Local Circuits in Primary Visual Cortex of the Macaque Monkey. *Annu. Rev. Neurosci.* 21: 47-74.

Calverley, R. K. S., K. S. Bedi, et al. (1988). Estimation of the numerical density of synapses in rat neocortex. Comparison of the `disector' with an `unfolding' method. *J. Neurosci. Meth.* 23(3): 195-205.

Calvin, W. H. (1995). Cortical Columns, Modules, and Hebbian Cell Assemblies. *The Handbook of Brain Theory and Neural Networks*. M. A. Arbib, Bradford Books / MIT Press**:** 269-272.

Carpenter, G. A. and S. Grossberg (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing* 37: 54-115.

Castro, H. A., S. M. Tam, et al. (1993). Implementation and performance of an analog nonvolatile neural network. *Analog Integrated Circuits and Signal Processing* 4(2): 97-113.

Catania, K. C., D. C. Lyon, et al. (1999). Cortical organization in shrews: Evidence from five species. *J. Comp. Neurol.* 410(1): 55-72.

CCortex (2003). *Press Release: Artificial Development to build world's biggest spiking neural network*,

Cestnik, B. (1990). Estimating Probabilities: A Crucial Task in Machine Learning. *In proc. of the Ninth European Conference on Artificial Intelligence (ECAI)*: 147-149.

Changeux, J.-P. (1985). *Neuronal Man: The Biology of Mind*, Oxford University Press.

Changizi, M. A. (2001). Principles underlying mammalian neocortical scaling. *Biol. Cybern.* 84(3): 207-215.

Charles, D. and C. Fyfe (1998). Modelling multiple-cause structure using rectification constraints. *Network: Comp. in Neur. Sys.* 9(2): 167-182.

Chen, H. A., Y. O. Carrasco, et al. (2000). *MPI Collective Operations over IP Multicast.* Personal Computer based Networks Of Workstations (PC-NOW 2000), Cancun, Mexico.

Chenn, A. and C. A. Walsh (2002). Regulation of Cerebral Cortical Size by Control of Cell Cycle Exit in Neural Precursors. *Science* 297(5580): 365-369.

Cherniak, C. (1995). Neural component placement. *Trends in Neurosci.* 18(12): 522-527.

Cherniak, C., Z. Mokhtarzada, et al. (2004). Global optimization of cerebral cortex layout. *Proc. Natl. Acad. Sci. USA* 101(4): 1081-1086.

Chicca, E., D. Badoni, et al. (2003). A VLSI recurrent network of integrate-and-fire neurons connected by plastic synapses with long-term memory. *IEEE Trans. on Circuits and Systems* 14(5): 1297-1307.

Chklovskii, D. B., T. Schikorski, et al. (2002). Wiring Optimization in Cortical Circuits. *Neuron* 34(3): 341-347.

Christos, G. A. (1996). Investigation of the Crick-Mitchison reverse-learning dream sleep hypothesis in a dynamical setting. *Neural Networks* 9(3): 427-434.

Chua, L. O. and L. Yang (1988). Cellular Neural Networks: Theory. *IEEE Trans. on Circuits and Systems* 35(10): 1257-1272.

Clark, D. A., P. P. Mitra, et al. (2001). Scalable architecture in mammalian brains. *Nature* 411(6834): 189-193.

Cohen, M. and S. Grossberg (1983). Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competetive Neural Networks. *IEEE Trans. Syst., Man, Cybern.* 13: 815-826

Conway, B., J. D. Boyd, et al. (2000). The Projection from V1 to Extrastriate Area 21a: A Second Patchy Efferent Pathway that Colocalizes with the CO Blob Columns in Cat Visual Cortex. *Cerebral Cortex* 10(2): 149-159.

Cook, J. (1989). The mean-field theory of a Q-state neural network model. *Physical Review A* 22(12): 2057-2067.

Cossart, R., D. Aronov, et al. (2003). Attractor dynamics of network UP states in the neocortex. *Nature* 423(6937): 283-288.

Culurciello, E. and A. G. Andreou (2003). A comparative study of access topologies for chip-level address-event communication channels. *IEEE Trans. on Neural Networks* 14(5): 1266- 1277.

Cürüklü, B. (2005). *A Canonical Model of the Primary Visual Cortex*. Ph. D. thesis, Dep. of Computer Science and Electronics, Mälardalen University, Västerås, Sweden.

Cürüklü, B. and A. Lansner (2002). *An Abstract Model of a Cortical Hypercolumn* 1: 80-85. International Conference on Neural Information Processing (ICONIP '02), Singapore, IEEE.

Dancause, N., S. Barbay, et al. (2005). Topographically Divergent and Convergent Connectivity between Premotor and Primary Motor Cortex. *Cerebral Cortex*.

Dara, R., S. C. Kremer, et al. (2002). *Clustering unlabeled data with SOMs improves classification oflabeled real-world data* 3: 2237-2242. International Joint Conference on Neural Networks (IJCNN '2002), Honolulu, HI, USA.

Davey, N., S. P. Hunt, et al. (2004). High capacity recurrent associative memories. *Neurocomputing* 62: 459-491.

Dayan, P., G. E. Hinton, et al. (1995). The Helmholtz Machine. *Neural Computation* 7: 889-904.

Dayan, P. and D. Willshaw (1991). Optimising Synaptic Learning Rules in Linear Associative Memories. *Biol. Cybern.* 65: 253-265.

Dayan, P. and R. S. Zemel (1995). Competition and Multiple Cause Models. *Neural Computation* 7: 565-579.

Dean, T. (2005). *A computational model of the cerebral cortex*: 938-943. AAAI-05, Cambridge, Massachusetts, MIT Press.

Deco, G. (2001). Top-down Selective Visual Attention: A Neurodynamical Approach. *Visual Cognition* 8(1): 119-140.

Deco, G. and T. S. Lee (2002). A unified model of spatial and object attention based on inter-cortical biased competition. *Neurocomputing* 44-46: 775-781.

Deco, G. and L. Parra (1997). Non-linear Feature Extraction by Redundancy Reduction in an Unsupervised Stochastic Neural Network. *Neural Networks* 10(4): 683-691.

Deco, G. and E. Rolls (2004). A Neurodynamical Cortical Model of Visual Attention and Invariant Object Recognition. *Vision Research* 44(6): 621-642.

Deco, G. and B. Schürmann (2000). A hierarchical neural system with attentional top-down enhancement of the spatial resolution for object recognition. *Vision Research* 40(20): 2845-2859.

DeFelipe, J., M. Conley, et al. (1986). Long-range focal collateralization of axons arising from corticocortical cells in monkey sensory-motor cortex. *J. Neurosci.* 6(12): 3749-3766.

DeFelipe, J. and I. Fariñas (1992). The pyramidal neuron of the cerebral cortex: Morphological and chemical characteristics of the synaptic inputs. *Prog. in Neurobio.* 39(6): 563-607.

DeFelipe, J., S. H. C. Hendry, et al. (1990). A microcolumnar structure of monkey cerebral cortex revealed by immunocytochemical studies of double bouquet cell axons. *Neuroscience* 37(3): 655-673.

DeFelipe, J., P. Marco, et al. (1999). Estimation of the Number of Synapses in the Cerebral Cortex: Methodological Considerations. *Cerebral Cortex* 9(7): 722-732.

Deiss, S. R., R. J. Douglas, et al. (1999). A Pulse-Coded Communication Infrastructure for Neuromorphic Systems. *Pulsed Neural Networks*. W. Maass and C. M. Bishop, MIT Press**:** 157-178.

Delorme, A., J. Gautrais, et al. (1999). SpikeNet: S Simulator for Modeling Large Networks of Integrate and Fire Neurons. *Neurocomputing* 26-27: 989-996.

Delorme, A. and S. Thorpe (2003). SpikeNet: an event-driven simulation package for modelling large networks of spiking neurons. *Network: Comp. in Neur. Sys.* 14: 613-627.

Diederich, S. and M. Opper (1987). Learning of correlated patterns in spin-glass networks by local learning rules. *Phys. Rev. Lett.* 58(9): 949-952.

Djurfeldt, M., C. Johansson, et al. (2005). *Massively parallel simulation of brain-scale neuronal network models*. TRITA-NA-P0513, Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Dombrowski, S., C. Hilgetag, et al. (2001). Quantitative Architecture Distinguishes Prefrontal Cortical Systems in the Rhesus Monkey. *Cerebral Cortex* 11(10): 975-988.

Dominguez, D. R. C. and D. Bollé (1998). Self-Control in Sparsely Coded Networks. *Phys. Rev. Lett.* 80(13): 2961-2964.

Dotsenko, V. S., N. D. Yarunin, et al. (1991). Statistical mechanics of Hopfield-like neural networks with modified interactions. *Journal of Physics A* 24(10): 2419-2429.

Douglas, R. J. and K. A. C. Martin (2004). Neuronal Circuits of the Neocortex. *Annu. Rev. Neurosci.* 27: 419-451.

Dow, B. M. (2002). Orientation and Color Columns in Monkey Visual Cortex. *Cerebral Cortex* 12(10): 1005-1015.

Draghici, S. (2000). Neural Networks in Analog Hardware - Design and Implementation Issues. *Int. J. Neural Systems* 19(1): 19-42.

Draghici, S. (2002). On the capabilities of neural networks using limited precision weights. *Neural Networks* 15(3): 395-414.

Eccles, J. C. (1981). The modular operation of the cerebral neocortex considered as the material basis of mental events. *Neuroscience* 6(10): 1839-1856.

Egmont-Petersen, M., D. d. Ridder, et al. (2002). Image processing with neural networks—a review. *Pattern Recognition* 35(10): 2279-2301.

Egorov, A. V., B. N. Hamam, et al. (2002). Graded persistent activity in entorhinal cortex neurons. *Nature* 420(6912): 173-178.

Elliffe, M. C. M., E. T. Rolls, et al. (2002). Invariant recognition of feature combinations in the visual system. *Biol. Cybern.* 86: 59-71.

Eriksson, D. and A. Lansner (2003). *Dynamic Hierarchical Attractor-Clustering*. TRITA-NA-P0304, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Feldman, J. A. and D. H. Ballard (1982). Connectionist Models and Their Properties. *Cognitive Science* 6(3): 205-254.

Feldman, M. L. and A. Peters (1974). A study of barrels and pyramidal dendritic clusters in the cerebral cortex. *Brain Res.* 77(1): 55-76.

Ferrari, P. A., S. Martínez, et al. (1992). A Lower Bound for the Memory Capacity in the Potts-Hopfield Model. *Journal of Statistical Physics* 66(5-6): 1643-1652.

Field, D. J. (1994). What Is the Goal of Sensory Coding. *Neural Computation* 6: 559-601.

Fischl, B. and A. M. Dale (2000). Measuring the thickness of the human cerebral cortex from magnetic resonance images. *Proc. Natl. Acad. Sci. USA* 97(20): 11050-11055.

Forum, M. P. I. (1995). *MPI: A Message-Passing Interface Standard - Message Passing Interface*, University of Tennessee, Knoxville.

Forum, M. P. I. (2003). *MPICH-A Portable Implementation of MPI*, University of Tennessee, Knoxville.

Fransén, E. and A. Lansner (1998). A model of cortical associative memory based on a horizontal network of connected columns. *Network: Comp. in Neur. Sys.* 9(2): 235-264.

Frey, U. and R. Morris (1997). Synaptic tagging - synapse specificity during protein synthesisdependent long-term potentiation. *Nature* 385(6616): 533-536.

Frolov, A., A. Kartashov, et al. (1995a). Quality and efficiency of retrieval for Willshaw-like autoassociative networks. I. Correction. *Network: Comp. in Neur. Sys.* 6(4): 513-534.

Frolov, A., A. Kartashov, et al. (1995b). Quality and efficiency of retrieval for Willshaw-like autoassociative networks. II. Recognition. *Network: Comp. in Neur. Sys.* 6(4): 535-549.

Frolov, A. A., D. Husek, et al. (1997). Informational Capacity and Recall Quality in Sparsely Encoded Hopfield-like Neural Network: Analytical Approaches and Computer Simulation. *Neural Networks* 10(5): 845-855.

Frolov, A. A. and I. P. Murav'ev (1993). Informational characteristics of neural networks capable of associative learning based on Hebbian plasticity. *Network: Comp. in Neur. Sys.* 4(4): 495-536.

Fukushima, K. (1988). A Neural Network for Visual Pattern Recognition. *Computer* 21(3): 65-75.

Fukushima, K. (1989). Analysis of the Process of Visual Pattern Recognition by the Neocognitron. *Neural Networks* 2(6): 413-420.

Fukushima, K. (1999). Self-organization of shift-invariant receptive fields *Neural Networks* 12(6): 791-801.

Fukushima, K. and N. Wake (1991). Handwritten Alphanumeric Character Recognition by the Neocognitron. *IEEE Trans. on Neural Networks* 2(3): 355-365.

Fulvi Mari, C. and A. Treves (1998). Modeling neocortical areas with a modular neural network. *BioSystems* 48: 47-55.

Fusi, S. (2001). Long term memory: Encoding and storing strategies of the brain. *Neurocomputing* 38-40: 1223-1228.

Fusi, S. (2002). Hebbian spike-driven synaptic plasticity for learning patterns of mean firing rates. *Biol. Cybern.* 87(5-6): 459-470.

Fusi, S., M. Annunziato, et al. (2000). Spike-Driven Synaptic Plasticity: Theory, Simulation, VLSI Implementation. *Neural Computation* 12(10): 2227-2258.

Fusi, S., P. Drew, et al. (2005). Cascade Models of Synaptically Stored Memories. *Neuron* 45: 599-611.

Fusi, S. and W. Senn (2006). Eluding oblivion with smart stochastic selection of synaptic updates. *Chaos* 16(026112): 1-11.

Földiak, P. (1990). Forming sparse representations by local anti-Hebbian learning. *Biol. Cybern.* 64: 165-170.

Földiák, P. (1991). Learning Invariance from Transformation Sequences. *Neural Computation* 3: 194-200.

Galuske, R. A. W., W. Schlote, et al. (2000). Interhemispheric Asymmetries of the Modular Structure in Human Temporal Cortex. *Science* 289(5486): 1946-1949.

Gara, A., M. A. Blumrich, et al. (2005). Overview of the Blue Gene/L system architecture. *IBM Journal of Reasearch and Development* 49(2/3): 195-212.

Gardner, E. (1987a). Maximum Storage Capacity in Neural Networks. *Europhysics Letters* 4(4): 481-485.

Gardner, E. (1987b). Multiconnected neural network models. *Journal of Physics A* 20(11): 3453-3464.

Gardner, E. (1988). The space of interactions in neural network models. *Journal of Physics A* 21(1): 257-270.

Gardner, E. J., D. J. Wallace, et al. (1989). Training with noise and the storage of correlated patterns in a neural network model. *Journal of Physics A* 22(12): 2019-2030.

Gayrard, V. (1992). Thermodynamic Limit of the q-State Potts-Hopfield Model with Infinitely Many Patterns. *Journal of Statistical Physics* 68(5-6): 977-1011.

Gazzaniga, M. S., R. B. Ivry, et al. (1998). *Cognitive Neuroscience: The Biology of the Mind*. New York, W. W. Norton & Company.

Geist, A., A. Beguelin, et al. (1994). *PVM: Parallel Virtual Machine, A Users' Guide and Tutorial for Networked Parallel Computing*. Cambridge, MA, MIT Press.

Gelman, A., J. B. Carlin, et al. (2004). *Bayesian Data Analysis*. Boca Raton, Florida, USA, Chapman & Hall/CRC.

George, D. and J. Hawkins (2005). *A Hierarchical Bayesian Model of Invariant Pattern Recognition in the Visual Cortex* 3: 1812-1817 International Joint Conference on Neural Networks (IJCNN '05), IEEE.

Gerstner, W. (1998). Populations of Spiking Neurons. *Pulsed Neural Networks*. W. Maass and C. M. Bishop, MIT press**:** 261-295.

Gerstner, W. and W. M. Kistler (2002). Mathematical formulations of Hebbian learning. *Biol. Cybern.* 87(5-6): 404-415.

Geszti, T. and F. Pazmandi (1987). Learning within bounds and dream sleep. *Journal of Physics A* 20(18): L1299-L1303.

Ghosh, J. and K. Hwang (1989). Mapping neural networks onto message-passing multicomputers. *Journal of Parallel and Distributed Computing* 6(2): 291-330.

Gilbert, C. D. and T. N. Wiesel (1989). Columnar specificity of intrinsic horizontal and corticocortical connections in cat visual cortex. *J. Neurosci.* 9(7): 2432-2442.

Girau, B. (2000). FPNA: Interaction Between FPGA and Neural Computation. *Int. J. Neural Systems* 10(3): 243-259.

Girman, S. V., Y. Sauvé, et al. (1999). Receptive Field Properties of Single Neurons in Rat Primary Visual Cortex. *J. Neurophysiol.* 82: 301-311.

Giudice, P. D., S. Fusi, et al. (1998). Learning attractors in an asynchronous, stochastic electronic neural network. *Network: Comp. in Neur. Sys.* 9(2): 183-205.

Goldman, P. S. and W. J. H. Nauta (1977). Columnar distribution of cortico-cortical fibers in the frontal association, limbic, and motor cortex of the developing rhesus monkey. *Brain Res.* 122(3): 393-413.

Golomb, D., N. Rubin, et al. (1990). Willshaw model: Associative memory with sparse coding and low firing rates. *Physical Review A* 41(4): 1843-1854.

González-Burgos, G., G. Barrionuevo, et al. (2000). Horizontal Synaptic Connections in Monkey Prefrontal Cortex: An In Vitro Electrophysiological Study. *Cerebral Cortex* 10(1): 82-92.

Graham, B. and D. Willshaw (1995). Improving Recall from an Associative Memory. *Biol. Cybern.* 72(4): 337-346.

Graham, B. and D. Willshaw (1997). Capacity and information efficiency of the associative net. *Network: Comp. in Neur. Sys.* 8(1): 35-54.

Grassmann, C. and J. K. Anlauf (1999). Fast Digital Simulation Of Spiking Neural Networks And Neuromorphic Integration With Spikelab. *Int. J. Neural Systems* 9(5): 473 - 478.

Grossberg, S. (1987). Competitive Learning: From Interactive Activation to Adaptive Resonance. *Cognitive Science* 11: 23-63.

Grossberg, S. (1988). Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural Networks* 1(1): 17-61.

Gurney, K. (1997). *An Introduction to Neural Networks*. London, UCL Press Ltd.

Hammarlund, P. (1996). *Techniques for Efficient Parallel Scientific Computing*. Ph. D. Thesis, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Hammerstrom, D. (1987). *The Connectivity Requirements of Simple Association, or How Many Connections Do You Need?*: 338-347. Conference on Neural Network Information Processing, IEEE

Hammerstrom, D. (1995). A Digital VLSI Architecture for Real World Applications. *An Introduction to Neural and Electronic Networks, Electronic Network*. S. F. Zornetzer, J. L. Davis, C. Lau and T. McKenna, Academic Press.

Hammerstrom, D. (2000). *Computational Neurobiology Meets Semiconductor Engineering*. Invited Paper Multi-Valued Logic Conference, Portland, Oregon.

Hammerstrom, D., C. Gao, et al. (2003). FPGA Implementation Of Very Large Associative Memories - Scaling Issues. *FPGA Implementations of Neural Networks, in press*. A. Omondi. Boston, Kluwer Academic Publishers.

Harpur, G. F. and R. W. Prager (1996). Development of low entropy coding in a recurrent network. *Network: Comp. in Neur. Sys.* 7(2): 277-284.

Harris, F. C., J. Baurick, et al. (2002). *A Novel Parallel Hardware and Software Solution for a Large-Scale Biologically Realistic Cortical Simulation*, Goodman Brain Computation Lab, University of Nevada,

Harrison, K. H., P. R. Hof, et al. (2002). Scaling laws in the mammalian neocortex: Does form provide clues to function? *J. Neurocytology* 31(3-5): 289-298.

Harvey, R. L. (1994). *Neural Network Principles*. Englewood Cliffs, New Jersey, Prentice-Hall Inc.

Haug, H. (1987). Brain Sizes, Surfaces, and Neuronal Sizes of the Cortex Cerebri: A Stereological Investigation of Man and His Variability and a Comparison with Some Mammals (Primates, Whales, Marsupials, Insectivores, and One Elephant). *Am. J. Anatomy* 180: 126-142.

Hawkins, J. (2004). *On Intelligence*, Times Books.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, New Jersey, USA, Prentice Hall.

Hebb, D. O. (1949). *The Organization of Behavior*. New York, Wiley.

Heimer, L. (1988). *The Human Brain and Spinal Cord: Functional Neuroanatomy and Dissection Guide*. New York, Springer-Verlag.

Hellwig, B. (2000). A quantitative analysis of the local connectivity between pyramidal neurons in layers 2/3 of the rat visual cortex. *Biol. Cybern.* 82(2): 111-121.

Hemmen, J. L. v. (1997). Hebbian learning, its correlation catastrophe, and unlearning. *Network: Comp. in Neur. Sys.* 8(3): V1-V17.

Hendrich, N. (1996). *A Scalable Architecture for Binary Couplings Attractor Neural Networks*. Conf. on Microelectronics for Neural Networks and Fuzzy Systems (MicroNeuro'96), Lausanne, Switzerland.

Henry, C. (2005). Basal metabolic rate studies in humans: measurement and development of new equations. *Public Health Nutr.* 8(Special Issue 1): 1133-1152.

Hertz, J., A. Krogh, et al. (1991). *Introduction to the Theory of Neural Computation*, Addison-Wesely.

Hille, B. (1992). *Ionic Channels of Excitable Membranes*. Sunderland, Massachusetts, Sinauer Associates Inc.

Hirase, H. and M. Recce (1996). A search for the optimal thresholding sequence in an associative memory. *Network: Comp. in Neur. Sys.* 7(4): 741-756.

Hochreiter, S. and J. Schmidhuber (1999). Feature Extraction Through LOCOCODE. *Neural Computation* 11(3): 679-714.

Hoehfeld, M. and S. E. Fahlman (1992). Learning with Limited Numerical Precision Using the Cascade-Correlation Algorithm. *IEEE Trans. on Neural Networks* 3(4): 602-611.

Hofman, M. A. (1985). Size and Shape of the Cerebral Cortex in Mammals: I. The Cortical Surface. *Brain Behav. Evol.* 27: 28-40.

Hofman, M. A. (1988). Size and Shape of the Cerebral Cortex in Mammals: II. The Cortical Volume. *Brain Behav. Evol.* 32: 17-26.

Holmgren, C., T. Harkany, et al. (2003). Pyramidal cell communication within local networks in layer 2/3 of rat neocortex. *Journal of Physiology* 551(1): 139-153.

Holst, A. (1997). *The Use of a Bayesian Neural Network Model for Classification Tasks*. Ph. D. Thesis, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Holt, J. L. and J.-N. Hwang (1993). Finite Precision Error Analysis of Neural Network Hardware Implementations. *IEEE Trans. on Computers* 42(3): 281-290.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* 79: 2554-2558.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci. USA* 81: 3088-3092.

Hopfield, J. J. (1990). The effectiveness of analogue 'neural network' hardware. *Network: Comp. in Neur. Sys.* 1(1): 27-40.

Hubel, D. H. and T. N. Wiesel (1963). Shape and Arrangement of Columns in Cat's Striate Cortex. *Journal of Physiology* 165: 559-568.

Hubel, D. H. and T. N. Wiesel (1977). Functional architecture of macaque monkey visual cortex. *Proc. R. Soc. Lond. B.* 198: 1-59.

Hupe, J. M., A. C. James, et al. (1998). Cortical feedback improves discrimination between figure and background by V1, V2 and V3 neurons. *Nature* 394(6695): 784-787.

Huttenlocher, P. R. (1979). Synaptic density in human frontal cortex — Developmental changes and effects of aging. *Brain Res.* 163(2): 195-205.

Hübner, M., D. Shoham, et al. (1997). Spatial Relationships among Three Columnar Systems in Cat Area 17. *J. Neurosci.* 17(23): 9270-9284.

Häkkinen, J., M. Lagerholm, et al. (1998). A Potts Neuron Approach to Communication Routing. *Neural Computation* 10(6): 1587-1599.

Häkkinen, J., M. Lagerholm, et al. (2000). Local Routing Algorithms Based on Potts Neural Networks. *IEEE Trans. on Neural Networks* 11(4): 970-977.

Indiveri, G. (2001). A neuromorphic VLSI device for implementing 2D selective attention systems. *IEEE Trans. on Neural Networks* 12(6): 1455-1463.

Ishii, S. and M.-a. Sato (1997). Chaotic Potts Spin Model for Combinatorial Optimization Problems. *Neural Networks* 10(5): 941-963.

Issa, N. P., C. Trepel, et al. (2000). Spatial Frequency Maps in Cat Visual Cortex. *J. Neurosci.* 20(22): 8504-8514.

Johannet, A., L. Personnaz, et al. (1992). Specification and Implementation of a Digital Hopfield-Type Associative Memory with On-Chip Training. *IEEE Trans. on Neural Networks* 3(4): 529-539.

Johansson, C. (2001a). *A Capacity Study of a Bayesian Neural Network with Hypercolumns*. TRITA-NA-P0120, Royal Institute of Technology, Dept. of Numerical Analysis and Computer Science, Stockholm, Sweden.

Johansson, C. (2001b). *A Study of Interacting Bayesian Recurrent Neural Networks with Incremental Learning*. M. Sc. Thesis, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Johansson, C. (2004). *Towards Cortex Isomorphic Attractor Neural Networks*. Licentiate Thesis, TRITA-NA-0415, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Johansson, C. and A. Lansner (2002a). *An Associative Neural Network Model of Classical Conditioning*. TRITA-NA-P0217, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Johansson, C. and A. Lansner (2002b). *A Neural Reinforcement Learning System*. TRITA-NA-P0215, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Johansson, C. and A. Lansner (2003). *Mapping of the BCPNN onto Cluster Computers*. TRITA-NA-P0305, Royal Institute of Technology, Dept. of Numerical Analysis and Computer Science, Stockholm, Sweden.

Johansson, C. and A. Lansner (2004a). *BCPNN Implemented with Fixed-Point Arithmetic*. TRITA-NA-P0403, Royal Institute of Technology, Dept. of Numerical Analysis and Computer Science, Stockholm, Sweden.

Johansson, C. and A. Lansner (2004b). Towards Cortex Sized Artificial Nervous Systems. *in Proc. Knowledge-Based Intelligent Information and Engineering Systems, KES'04* LNAI 3213: 959-966.

Johansson, C. and A. Lansner (2004c). *Towards Cortex Sized Artificial Nervous Systems*. KES 2004, WelTec, Wellington, New Zealand.

Johansson, C. and A. Lansner (2004d). *Towards Cortex Sized Attractor ANN*: 383-398. Bio-ADIT 2004, EPFL, Lausanne, Switzerland, Springer-Verlag.

Johansson, C. and A. Lansner (2005). *A Mean Field Approximation of BCPNN*. TRITA-NA-P0506, Royal Institute of Technology, Dept. of Numerical Analysis and Computer Science, Stockholm, Sweden.

Johansson, C., P. Raicevic, et al. (2003). *Reinforcement Learning Based on a Bayesian Confidence Propagating Neural Network*. SAIS-SSLS Joint Workshop, Center for Applied Autonomous Sensor Systems, Örebro, Sweden.

Johansson, C. and M. Rehn (2004). *Small-World Connectivity and Attractor Neural Networks*, Royal Institute of Technology, Dept. of Numerical Analysis and Computer Science, Stockholm, Sweden.

Jolivet, R., T. J. Lewis, et al. (2004). Generalized Integrate-and-Fire Models of Neuronal Activity Approximate Spike Trains of a Detailed Model to a High Degree of Accuracy. *Journal of Neurophysiology* 92(2): 959-976.

Jones, E. G. (2000). Microcolumns in the cerebral cortex. *Proc. Natl. Acad. Sci. USA* 97(10): 5019-5021.

Jonsson, J. (2003). *Pilot Study of a Parallel Implementation of a Bayesian Confidence Propagating Neural Network*. M. Sc. Thesis, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Jouve, B., P. Rosenstiehl, et al. (1998). A mathematical approach to the connectivity between the cortical visual areas of the macaque monkey. *Cerebral Cortex* 8(1): 28-39.

Jutten, C. and J. Herault (1991). Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture *Signal Processing* 24(1): 1-10.

Kaas, J. H. (1987). The Organization of Neocortex in Mammals: Implications for Theories of Brain Function. *Annu. Rev. Psychol.* 38: 129-151.

Kaas, J. H. (2000). Why is Brain Size so Important: Design Problems and Solutions as Neocortex Gets Bigger or Smaller. *Brain and Mind* 1(1): 7-23.

Kamimura, R. (2005). Unifying cost and information in information-theoretic competitive learning. *Neural Networks* 18(5-6): 711-718.

Kandel, E. R., J. H. Schwartz, et al. (2000). *Principles of Neural Science*. New York, McGraw-Hill.

Kanter, I. (1988). Potts-glass models of neural networks. *Physical Review A* 37(7): 2739-2742.

Kanter, I. and H. Sompolinsky (1987). Associative recall of memory without errors. *Physical Review A* 35(1): 380–392.

Karayiannis, N. B. and A. N. Venetsanopoulos (1990). On the dynamics of neural networks realizing associative memories of first and higher order. *Network: Comp. in Neur. Sys.* 1(3): 345-364.

Karbowski, J. (2003). How Does Connectivity Between Cortical Areas Depend on Brain Size? Implications for Efficient Computation. *Journal of Computational Neuroscience* 15(3): 347-356.

Karhunen, J., E. Oja, et al. (1997). A class of neural networks for independent component analysis. *IEEE Trans. on Neural Networks* 8(3): 486-504.

Kartashov, A., A. Frolov, et al. (1997). Quality and efficiency of retrieval for Willshaw-like autoassociative networks: III. Willshaw - Potts model. *Network: Comp. in Neur. Sys.* 8(1): 71-86.

Kawaguchi, Y. and Y. Kubota (1997). GABAergic cell subtypes and their synaptic connections in rat frontal cortex. *Cerebral Cortex* 7(6): 476-486.

Kenan-Vaknin, G., G. E. Ouaknine, et al. (1992). Organization of layers II–III connections in human visual cortex revealed by in vitro injections of biocytin. *Brain Res.* 594(2): 339-342.

Kenet, T., D. Bibitchkov, et al. (2003). Spontaneously emerging cortical representations of visual attributes. *Nature* 425(6961): 954-956.

Kim, B. J. (2004). Performance of networks of artificial neurons: The role of clustering. *Phys. Rev. E* 69(4): 045101:1-4.

Kisvarday, Z., E. Toth, et al. (1997). Orientation-specific relationship between populations of excitatory and inhibitory lateral connections in the visual cortex of the cat. *Cerebral Cortex* 7(7): 605-618.

Klyachko, V. A. and C. F. Stevens (2003). Connectivity optimization and the positioning of cortical areas. *Proc. Natl. Acad. Sci. USA* 100(13): 7937-7941.

Koch, C. (1999). *Biophysics of Computation: Information Processing in Single Neurons*. New York, Oxford University Press.

Kononenko, I. (1989). Bayesian Neural Networks. *Biol. Cybern.* 61: 361-370.

Korbo, L., B. Pakkenberg, et al. (1990). An efficient method for estimating the total number of neurons in rat brain cortex. *J. Neurosci. Meth.* 31(2): 93-100.

Kraskov, A., H. Stögbauer, et al. (2005). Hierarchical clustering using mutual information. *Europhysics Letters* 70(2): 278-284.

Krauth, W. and M. Mezard (1987). Learning algorithms with optimal stability in neural networks. *Journal of Physics A* 20(11): L745-L752.

Kropff, E. and A. Treves (2005). The storage capacity of Potts models for semantic memory retrieval. *J. Stat. Mech.*: P08010.

Krubitzer, L. (1995). The organization of neocortex in mammals: are species differences really so different? *Trends in Neurosci.* 18(9): 408-417.

Kung, S. Y. and J. N. Hwang (1988). *Parallel Architectures for Artificial Neural Nets*: 165-172. International Conference on Neural Networks (ICNN'88), IEEE.

Körding, K. P. and D. M. Wolpert (2004). Bayesian integration in sensorimotor learning. *Nature* 427(6971): 244-247.

Lagerholm, M., C. Peterson, et al. (2000). Airline crew scheduling using Potts mean field techniques. *Euro. J. Operational Research* 120: 81-96.

Lamme, V. A., H. Supèr, et al. (1998). Feedforward, horizontal, and feedback processing in the visual cortex *Current Opinion in Neurobiology* 8(4): 529-535.

Lansner, A. and Ö. Ekeberg (1987). *An Associative Network Solving the "4-Bit ADDER Problem"* 2: 549-556. International Conference on Neural Networks (ICNN'87), San Diego.

Lansner, A. and Ö. Ekeberg (1989). A one-layer feedback artificial neural network with a Bayesian learning rule. *Int. J. Neural Systems* 1(1): 77-87.

Lansner, A. and E. Fransén (1992). Modeling Hebbian cell assemblies comprised of cortical neurons. *Network: Comp. in Neur. Sys.* 3: 105-119.

Lansner, A., E. Fransén, et al. (2003). Cell assembly dynamics in detailed and abstract attractor models of cortical associative memory. *Theory in Biosciences* 122: 19-36.

Lansner, A. and A. Holst (1996). A higher order Bayesian neural network with spiking units. *Int. J. Neural Systems* 7(2): 115-128.

Laughlin, S. B., R. R. d. R. v. Steveninck, et al. (1998). The metabolic cost of neural information. *Nature Neurosci.* 1(1): 36-41.

Lecun, Y., L. Bottou, et al. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE* 86(11): 2278-2324.

Lee, T. S. and D. Mumford (2003). Hierarchical Bayesian inference in the visual cortex. *J. Opt. Soc. Am. A Opt. Image. Sci. Vis.* 20(7): 1434-1448.

Lee, T. S. and M. Nguyen (2001). Dynamics of subjective contour formation in the early visual cortex. *Proc. Natl. Acad. Sci. USA* 98(4): 1907-1911.

Lehmann, C., M. Viredaz, et al. (1993). A generic systolic array building block for neural networks with on-chip learning. *IEEE Trans. on Neural Networks* 4(3): 400-407.

Leise, E. M. (1990). Modular construction of nervous systems: a basic principle of design for invertebrates and vertebrates. *Brain Res. Rev.* 15: 1-23.

Lennie, P. (2003). The Cost of Cortical Computation. *Current Biology* 13: 493-497.

Levin, B. (1995). *On Extensions, Parallel Implementation and Applications of a Bayesian Neural Network*. Ph. D. Thesis, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Levy, N., D. Horn, et al. (1999). Associative Memory in a Multimodular Network. *Neural Computation* 11(7): 1717-1737.

Li, C. and G. Chen (2003). Stability of a neural network model with small-world connections. *Phys. Rev. E* 68(5): 052901.

Likharev, K., A. Mayr, et al. (2003). CrossNets: high-performance neuromorphic architectures for CMOL circuits. *Ann. N Y Acad. Sci.* 1006(1): 146-163.

Linsker, R. (1986a). From basic network principles to neural architecture: Emergence of orientation-selective cells. *Proc. Natl. Acad. Sci.* 83: 8390-8394.

Linsker, R. (1986b). From basic network principles to neural architecture: Emergence of orientation columns. *Proc. Natl. Acad. Sci.* 83: 8779-8783.

Linsker, R. (1986c). From basic network principles to neural architecture: Emergence of spatial-opponent cells. *Proc. Natl. Acad. Sci.* 83: 7508-7512.

Linsker, R. (1988). Self-organization in a perceptual network. *IEEE Computer* 21: 105-117.

Liou, C.-Y. and J.-M. Wu (1996). Self-organization Using Potts Models. *Neural Networks* 9(4): 671-684.

Little, W. A. and G. L. Shaw (1975). A statistical theory of short and long term memory. *Behavioral Biology* 14: 115-133.

Liu, S.-C., J. Kramer, et al. (2001). Orientation-selective aVLSI spiking neurons. *Neural Networks* 14(6-7): 629-643.

Liu, S. and R. Douglas (2004). Temporal Coding in a Silicon Network of Integrate-and-Fire Neurons. *IEEE Trans. on Neural Networks* 15(5): 1305-1314.

London, M., A. Schreibman, et al. (2002). The information efficacy of a synapse. *Nature Neurosci.* 5(4): 332-340.

Luecke, G. R., B. Raffin, et al. (2000). Comparing the Communication Performance and Scalability of a Linux and an NT Cluster of PCs, a Cray Origin 2000, an IBM SP and a Cray

T3E-600. *Journal of Performance Evaluation and Modelling for Computer Systems (PEMCS).*

Lund, J. S., A. Angelucci, et al. (2003). Anatomical Substrates for Functional Columns in Macaque Monkey Primary Visual Cortex. *Cerebral Cortex* 13(1): 15-24.

Lundqvist, M., M. Rehn, et al. (2006a). Attractor dynamics in a modular network model of the neocortex. *Network: Comp. in Neur. Sys.* accepted.

Lundqvist, M., M. Rehn, et al. (2006b). Attractor dynamics in a modular network model of the cerebral cortex. *Neurocomputing* 69(10-12): 1155-1159.

Lücke, J. (2004). Hierarchical self-organization of minicolumnar receptive fields *Neural Networks* 17(8-9): 1377-1389.

Lücke, J. and C. v. d. Malsburg (2004). Rapid Processing and Unsupervised Learning in a Model of the Cortical Macrocolumn. *Neural Computation* 16(3): 501-533.

Maass, W. (2001). Neural computation: a research topic for theoretical computer science? Some thoughts and pointers. *Current Trends in Theoretical Computer Science, Entering the 21th Century*. G. Rozenberg, A. Salomaa and G. Paun, World Scientific Publishing**:** 680-690.

Mahapatra, S. and R. N. Mahapatra (2000). Mapping of Neural Network Models onto Systolic Arrays. *Journal of Parallel and Distributed Computing* 60: 677-689.

Malach, R., Y. Amir, et al. (1993). Relationship Between Intrinsic Connections and Functional Architecture Revealed by Optical Imaging and in vivo Targeted Biocytin Injections in Primate Striate Cortex. *Proc. Natl. Acad. Sci. USA* 90(22): 10469-10473.

Malenka, R. C. and a. R. A. Nicoll (1999). Long-Term Potentiation--A Decade of Progress? *Science* 285(5435): 1870-1874.

Maravall, M. (1999). Sparsification from dilute connectivity in a neural network model of memory. *Network: Comp. in Neur. Sys.* 10(1): 15-39.

Margaritis, K. G. (1995). On the systolic implementation of associative memory artificial neural networks. *Parallel Computing* 21: 825-840.

Markram, H. (2006). The Blue Brain Project. *Nature Rev. Neurosci.* 7(2): 153-160

Markram, H., M. Toledo-Rodriguez, et al. (2004). Interneurons of the neocortical inhibitory system. *Nature Rev. Neurosci.* 5(10): 793-807.

Marr, D. (1982). *Vision*. San Francisco.

Marshall, J. A. (1995). Adaptive perceptual pattern recognition by self-organizing neural networks: Context, uncertainty, multiplicity, and scale. *Neural Networks* 8(3): 335-362.

Martin, K. A. C. (2002). Microcircuits in visual cortex. *Current Opinion in Neurobiology* 12(4): 418-425.

Masuda, N. and K. Aihara (2004). Global and local synchrony of coupled neurons in small-world networks. *Biol. Cybern.* 90: 302-309.

Mattia, M. and P. D. Giudice (2000). Efficient Event-Driven Simulation of Large Networks of Spiking Neurons and Dynamical Synapses. *Neural Computation* 12: 2305-2329.

Mayhew, T. M., G. L. M. Mwamengele, et al. (1990). Comparative morphometry of the mammalian brain: estimates of cerebral volumes and cortical surface areas obtained from macroscopic slices. *Journal of Anatomy* 172: 191-200.

McCulloch, W. S. and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5: 115-133.

McEliece, R., E. Posner, et al. (1987). The capacity of the Hopfield associative memory. *IEEE Trans. on Information Theory* 33(4): 461- 482.

McGraw, P. N. and M. Menzinger (2003). Topology and computational performance of attractor neural networks. *Phys. Rev. E* 68(4): 047102.

Medler, D. A. (1998). A Brief History of Connectionism. *Neural Computing Surveys* 1: 61-101.

Mehrtash, N., D. Jung, et al. (2003). Synaptic plasticity in spiking neural networks (SP2INN): a system approach. *IEEE Trans. on Neural Networks* 14(5): 980-992.

Melton, M., T. Phan, et al. (1992). The TInMANN VLSI Chip. *IEEE Trans. on Neural Networks* 3(3): 375-384.

Mertens, S. (1991). An extremely diluted asymmetric network with graded response neurons. *Physical Review A* 24(1): 337-351.

Meunier, C., H.-F. Yanai, et al. (1991). Sparsely coded associative memories: capacity and dynamical properties. *Network: Comp. in Neur. Sys.* 2(4): 469-487.

Mézard, M., J. P. Nadal, et al. (1986). Solvable models of working memories. *J. Physique* 47: 1457-1462.

Michaels, R. (1996). Associative Memory with Uncorrelated Inputs. *Neural Computation* 8: 256-259.

Miki, T., Y. Fukui, et al. (1997). Estimation of the numerical densities of neurons and synapses in cerebral cortex. *Brain Res. Prot.* 2: 9-16.

Misra, M. (1997). Parallel Environments for Implementing Neural Networks. *Neural Computing Surveys* 1: 48-60.

Mitchell, T. M. (1997). *Machine Learning*, McGraw-Hill Inc.

Mitchison, G. (1992). Axonal trees and cortical architecture. *Trends in Neurosci.* 15(4): 122-126.

Montgomery, J. M. and D. V. Madison (2004). Discrete synaptic states define a major mechanism of synapse plasticity *Trends in Neurosci.* 27(12): 744-750

Mooney, S. M., R. M. A. Napper, et al. (1996). Long-Term Effect of Postnatal Alcohol Exposure on the Number of Cells in the Neocortex of the Rat: A Stereological Study. *Alcoholism: Clinical and Exper. Res.* 20(4): 615-623.

Morelli, L. G., G. Abramson, et al. (2004). Associative memory on a small-world neural network. *Eur. Phys. J. B* 38(3): 495-500.

Morita, M. (1993). Associative memory with nonmonotone dynamics. *Neural Networks* 6(1): 115-126.

Morrison, A., C. Mehring, et al. (2005). Advancing the Boundaries of High-Connectivity Network Simulation with Distributed Computing. *Neural Computation* 17(8): 1776-1801.

Mortara, A. and E. A. Vittoz (1994). A communication architecture tailored for analog VLSI artificial neural networks: intrinsic performance and limitations. *IEEE Trans. on Neural Networks* 5(3): 459-466.

Mountcastle, V. B. (1957). Modality and Topographic Properties of Single Neurons of Cat's Somatic Sensory Cortex. *Journal of Neurophysiology* 20: 408-434.

Mountcastle, V. B. (1997). The columnar organization of the neocortex. *Brain* 120: 701-722.

Mountcastle, V. B. (2003). Introduction (special issue on ‘Computation in Cortical Columns’). *Cerebral Cortex* 13(1): 2-4.

Mouton, P. R., D. L. Price, et al. (1997). Empirical assessment of synapse numbers in primate neocortex. *J. Neurosci. Meth.* 75(2): 119-126.

Muezzinoglu, M. K., C. Guzelis, et al. (2003). A new design method for the complex-valued multistate Hopfield associative memory. *IEEE Trans. on Neural Networks* 14(4): 891-899.

Muller, U. A., A. Gunzinger, et al. (1995). Fast neural net simulation with a DSP processor array. *IEEE Trans. on Neural Networks* 6(1): 203-213.

Murphy, K. M., D. G. Jones, et al. (1995). Cytochrome-Oxidase Blobs in Cat Primary Visual Cortex. *The Journal of Neuroscience* 15(6): 4196-4208.

Murray, S. O., D. Kersten, et al. (2002). Shape perception reduces activity in human primary visual cortex *Proc. Natl. Acad. Sci. USA* 99(23): 15164-15169.

Murray, S. O., P. Schrater, et al. (2004). Perceptual grouping and the interactions between visual cortical areas. *Neural Networks* 17(5-6): 695-705.

Murre, J. M. J. (1993). Transputers and Neural Networks: An Analysis of Implementation Constraints and Performance. *IEEE Trans. on Neural Networks* 4(2): 284-292.

Murre, J. M. J. and D. P. F. Sturdy (1995). The connectivity of the brain: multi-level quantitative analysis. *Biol. Cybern.* 73(6): 529-545.

Nadal, J.-P. and A. Rau (1991). Storage Capacity of a Potts-perceptron. *J. Phys. I France* 1: 1109-1121.

Nadal, J.-P. and G. Toulouse (1990). Information storage in sparsely coded memory nets. *Network: Comp. in Neur. Sys.* 1(1): 61-74.

Nadal, J. P., G. Toulouse, et al. (1986). Networks of Formal Neurons and Memory Palimpsests. *Europhysics Letters* 1(10): 535-542.

Nelson, M. E. and J. M. Bower (1990). Brain maps and parallel computers. *Trends in Neurosciences* 13(10): 403-408.

Nene, S. A., S. K. Nayar, et al. (1996). *Columbia Object Image Library (COIL-100)*. CUCS-006-96, Columbia Automated Vision Environment,

Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review* 45(2): 167-256.

Nicoll, A. and C. Blakemore (1993). Patterns of Local Connectivity in the Neocortex. *Neural Computation* 5(5): 665-680.

Nieuwenhuys, R., H. J. t. Donkelaar, et al. (1997). Section 22.11.6.6; Neocortex: Quantitative Aspects and Folding. *The Central Nervous System of Vertebrates*. Heidelberg, Springer-Verlag. 3**:** 2008-2013.

Nigam, K., A. K. Mccallum, et al. (2000). Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning* 39(2-3): 103 - 134

Noest, A. J. (1989). Discrete-state phasor neural networks. *Physical Review A* 38(4): 2196-2199.

Nordström, T. and B. Svensson (1992). Using and designing massively parallel computers for artificial neural networks. *Journal of Parallel and Distributed Computing* 14(3): 260-285.

Northcutt, R. G. and J. H. Kaas (1995). The emergence and evolution of mammalian neocortex. *Trends in Neurosci.* 18(9): 373-379.

O'Kane, D. and D. Sherrington (1993). A feature retrieving attractor neural network. *Journal of Physics A* 26(10): 2333-2342.

O'Kane, D. and A. Treves (1992a). Short- and long-range connections in autoassociative memory. *Journal of Physics A* 25(19): 5055-5069.

O'Kane, D. and A. Treves (1992b). Why the simplest notion of neocortex as an autoassociative memory would not work. *Network: Comp. in Neur. Sys.* 3(4): 379-384.

Obradovic, D. and G. Deco (1998). Information Maximization and Independent Component Analysis: Is there a Difference? *Neural Computation* 10(8): 2085-2101.

Okada, M. (1996). Notions of Associative Memory and Sparse Coding. *Neural Networks* 9(8): 1429-1458.

Olshausen, B. A. and D. J. Field (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381(6583): 607-609.

Olshausen, B. A. and D. J. Field (1997). Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1. *Vision Research* 37(23): 3311-3325.

Olshausen, B. A. and D. J. Field (2004). Sparse coding of sensory inputs. *Current Opinion in Neurobiology* 14: 481-487.

Oppenheim, A. V. and A. S. Willsky (1997). *Signals & Systems*. London, Prentice-Hall International.

Oram, M. W. and P. Földiák (1996). Learning generalisation and localisation: Competition for stimulus type and receptive field. *Neurocomputing* 11(2-4): 287-321.

Orre, R. (1998). *Data Mining and Process Modelling using a Bayesian Confidence Propagation Neural Network*. Lic. Thesis, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Orre, R., A. Bate, et al. (2005). A BAYESIAN RECURRENT NEURAL NETWORK FOR UNSUPERVISED PATTERN RECOGNITION IN LARGE INCOMPLETE DATA SETS. *Int. J. Neural Systems* 15(3): 207-222.

Orre, R., A. Lansner, et al. (2000). Bayesian neural networks with confidence estimations applied to data mining. *Computational Statistics and Data Analysis* 34(8): 473-493.

Pakkenberg, B. and H. J. G. Gundersen (1997). Neocortical Neuron Number in Humans: Effect of Sex and Age. *J. Comp. Neurol.* 384: 312-320.

Pakkenberg, B., D. Pelvig, et al. (2003). Aging and the human neocortex. *Experimental Gerontology* 38: 95-99.

Palm, G. (1980). On Associative Memory. *Biol. Cybern.* 36(1): 19-31.

Palm, G. (1981a). On the Storage Capacity of an Associative Memory with Randomly Distributed Storage Elements. *Biol. Cybern.* 39(2): 125-127.

Palm, G. (1981b). Towards a Theory of Cell Assemblies. *Biol. Cybern.* 39(3): 181-194.

Palm, G. (1982). *Neural Assemblies: An Alternative Approach to Artificial Intelligence*. New York, Springer-Verlag.

Palm, G. and F. T. Sommer (1992). Information capacity in recurrent McCulloch-Pitts networks with sparsely coded memory states. *Network: Comp. in Neur. Sys.* 3(2): 177-186.

Parga, N. and E. Rolls (1998). Transform-Invariant Recognition by Association in a Recurrent Network. *Neural Computation* 10(6): 1507-1525.

Parisi, G. (1986a). Asymmetric neural networks and the process of learning. *Journal of Physics A* 19(11): L675-L680.

Parisi, G. (1986b). A memory which forgets. *Journal of Physics A* 19(10): L617-L620.

Peper, F. and M. N. Shirazi (1996). A Categorizing Associative Memory Using an Adaptive Classifier and Sparse Coding. *IEEE Trans. on Neural Networks* 7(3): 669-675.

Peretto, P. (1984). Collective Properties of Neural Networks: A Statistical Physics Approach. *Biol. Cybern.* 50: 51-62.

Personnaz, L., I. Guyon, et al. (1986). Collective computational properties of neural networks: New learning mechanisms. *Physical Review A* 34(5): 4217–4228.

Peters, A. and D. A. Kara (1987). The Neuronal Composition of Area 17 of Rat Visual Cortex. IV. The Organization of Pyramidal Cells. *J. Comp. Neurol.* 260: 573-590.

Peters, A. and C. Sethares (1991). Organization of Pyramidal Neurons in Area 17 of Monkey Visual Cortex. *J. Comp. Neurol.* 306: 1-23.

Peters, A. and C. Sethares (1996). Myelinated axons and the pyramidal cell modules in monkey visual cortex. *J. Comp. Neurol.* 365: 232-255.

Peters, A. and E. Yilmaz (1993). Neuronal Organization in Area 17 of Cat Visual Cortex. *Cerebral Cortex* 3(1): 49-68.

Petersen, C. C., R. C. Malenka, et al. (1998). All-or-none potentiation at CA3-CA1 synapses. *Proc. Natl. Acad. Sci. USA* 95(8): 4732-4737.

Peterson, C. and B. Söderberg (1989). A New Method for Mapping Optimization Problems onto Neural Networks. *Int. J. Neural Systems* 1(1): 3-22.

Price, D. J., J. M. Ferrer, et al. (1994). Functional organization of corticocortical projections from area 17 to area 18 in the cat's visual cortex. *The Journal of Neuroscience* 14(5): 2732-2746.

Rakic, P. (1974). Neurons in the rhesus monkey visual cortex: systematic relation between time of origin and eventual disposition. *Science* 183: 425-427.

Rakic, P. (1995). A small step for the cell, a giant leap for mankind: a hypothesis of neocortical expansion during evolution. *Trends in Neurosci.* 18(9): 383-388.

Rao, A. R., G. A. Cecchi, et al. (2005). *Unsupervised segmentation with dynamical units*, IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY.

Rao, R. P. N. and D. H. Ballard (1997). Dynamic Model of Visual Recognition Predicts Neural Response Properties in the Visual Cortex. *Neural Computation* 9: 721-763.

Rao, R. P. N. and D. H. Ballard (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neurosci.* 2(1): 79-87.

Rehn, M. and F. Sommer (2006). Storing and restoring visual input with collaborative rank coding and associative memory. *Neurocomputing* 69(10-12): 1219-1223.

Renart, A., N. Parga, et al. (1999). Backprojections In The Cerebral Cortex: Implications For Memory Storage. *Neural Computation* 11(6): 1349-1388.

Reutimann, J., M. Giugliano, et al. (2003). Event-Driven Simulation of Spiking Neurons with Stochastic Dynamics. *Neural Computation* 15(4): 811-830.

Rieger, H. (1990). Storing an extensive number of grey-toned patterns in a neural network using multistate neurons. *Physical Review A* 23(23): L1273-L1279.

Ringach, D. L., M. J. Hawken, et al. (1997). Dynamics of orientation tuning in macaque primary visual cortex. *Nature* 387(6630): 281-284.

Robins, A. and S. McCallum (1999). The consolidation of learning during sleep: comparing the pseudorehearsal and unlearning accounts. *Neural Networks* 12(7-8): 1191-1206.

Robins, A. V. and S. J. R. McCallum (2004). A robust method for distinguishing between learned and spurious attractors. *Neural Networks* 17(2): 313-326.

Rockel, A. J., R. W. Hiorns, et al. (1980). The Basic Uniformity in Structure of the Neocortex. *Brain* 103: 221-244.

Rolls, E. T., N. C. Aggelopoulos, et al. (2004). Information encoding in the inferior temporal visual cortex: contributions of the firing rates and the correlations between the firing of neurons. *Biol. Cybern.* 90(1): 19-32.

Rolls, E. T. and A. Treves (1998). *Neural Networks and Brain Function*. New York, Oxford University Press.

Rubner, J. and K. Schulten (1990). Development of feature detectors by self-organization: A network model *Biol. Cybern.* 62(3): 193-199.

Rumelhart, D. E., G. E. Hinton, et al. (1986). Learning representations by back-propagating errors. *Nature* 323: 533-536.

Rumelhart, D. E. and D. Zipser (1985). Feature Discovery by Competetive Learning. *Cognitive Science* 9: 75-112.

Ruthazer, E. S. and M. P. Stryker (1996). The Role of Activity in the Development of Long-Range Horizontal Connections in Area 17 of the Ferret *The Journal of Neuroscience* 16(22): 7253-7269.

Salin, P.-A. and J. Bullier (1995). Corticocortical Connections in the Visual System: Structure and Function. *Physiological Reviews* 75(1): 107-154.

Sandberg, A. (2003). *Bayesian Attractor Neural Network Models of Memory*. Ph. D. Thesis, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Sandberg, A. and A. Lansner (2001). Selective enhancement of recall through plasticity modulation in an autoassociative memory. *Neurocomputing* 38-40: 867-873.

Sandberg, A. and A. Lansner (2002). Synaptic depression as an intrinsic driver of reinstatement dynamics in an attractor network. *Neurocomputing* 44-46: 615-622.

Sandberg, A., A. Lansner, et al. (2000). A palimpsest memory based on an incremental Bayesian learning rule. *Neurocomputing* 32-33: 987-994.

Sandberg, A., A. Lansner, et al. (2002). A Bayesian attractor network with incremental learning. *Network: Comp. in Neur. Sys.* 13(2): 179-194.

Sandberg, A., J. Tegnér, et al. (2003). A Working Memory Model Based on Fast Learning. *Network: Comp. in Neur. Sys.* 14(4): 789-802.

Sandh, K. (2004). *Studie av BCPNN med gles kopplingsmatris*. M. Sc. Thesis, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Saund, E. (1995). A Multiple Cause Mixture Model for Unsupervised Learning. *Neural Computation* 7(1): 51-71.

Scannell, J. W. (1997). Determining cortical landscapes. *Nature* 386(6624): 452.

Scannell, J. W., C. Blakemore, et al. (1995). Analysis of Connectivity in the Cat Cerebral Cortex. *The Journal of Neuroscience* 15(2): 1463-1483.

Scannell, J. W., G. A. P. C. Burns, et al. (1999). The Connectional Organization of the Cortico-thalamic System of the Cat. *Cerebral Cortex* 9(3): 277-299.

Schæfer, M., T. Schnauer, et al. (2002). Simulation of spiking neural networks — architectures and implementations. *Neurocomputing* 48(1-4): 647-679.

Scheff, S. W., S. T. DeKosky, et al. (1990). Quantitative assessment of cortical synaptic density in Alzheimer's disease. *Neurobiology of Aging* 11(1): 29-37.

Scheff, S. W., D. A. Price, et al. (2001). Quantitative assessment of possible age-related change in synaptic numbers in the human frontal cortex. *Neurobiology of Aging* 22(3): 355-365.

Schmidhuber, J. (1992). Learning Factorial Codes by Predicatability Minimization. *Neural Computation* 4: 863-879.

Schmitt, M. (2001). On the Complexity of Computing and Learning with Multiplicative Neural Networks.

Schoenauer, T., S. Atasoy, et al. (2002). NeuroPipe-Chip: A digital neuro-processor for spiking neural networks. *IEEE Trans. on Neural Networks* 13(1): 205-213.

Schraudolph, N. N. and T. J. Sejnowski (1992). Competitive Anti-Hebbian Learning of Invariants. *Advances of Information Processing Systems* 4: 1017-1024.

Schubert, D., J. F. Staiger, et al. (2001). Layer-Specific Intracolumnar and Transcolumnar Functional Connectivity of Layer V Pyramidal Cells in Rat Barrel Cortex. *The Journal of Neuroscience* 21(10): 3580-3592.

Schwenker, F., F. T. Sommer, et al. (1996). Iterative retrieval of sparsely coded associative memory patterns. *Neural Networks* 9(3): 445-455.

Schüz, A. and H. Preibl (1996). Basic Connectivity of the Cerebral Cortex and some Considerations on the Corpus Callosum. *Neuroscience & Biobehavioral Reviews* 20(4): 567-570.

Seiffert, U. (2004). Artificial Neural Networks on Massively Parallel Computer Hardware. *Neurocomputing* 57: 135-150.

Semendeferi, K., E. Armstrong, et al. (2001). Prefrontal cortex in humans and apes: A comparative study of area 10. *Am. J. Phys. Anthropol.* 114(3): 224-241.

Serbedzija, N. (1996). Simulating Artificial Neural Networks on Parallel Architectures. *Computer* 29(3): 56-63.

Shakorian, A. (2005). *Broadcast Communication in Parallel Implementations of Cortical Sized Neural Networks*. Master Thesis, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Master Thesis, TRITA-NA-E05057, Stockholm, Sweden.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal* 27: 379-423 and 623-656.

Shepherd, G. M. and C. Koch (1998). *The Synaptic Organization of the Brain*, Oxford University Press.

Shiino, M. and T. Fukai (1993). Self-consistent signal-to-noise analysis of the statistical behavior of analog neural networks and enhancement of the storage capacity. *Phys. Rev. E* 48(2): 867-897.

Shim, G. M., D. Kim, et al. (1991). Statistical-mechanical formulation of the Willshaw model with local inhibition. *Physical Review A* 43(12): 7012-7018.

Shim, G. M., D. Kim, et al. (1992). Potts-glass model of layered feedforward neural networks. *Physical Review A* 45(2): 1238-1248.

Shinomoto, S. (1987). Memory maintenance in neural networks. *Journal of Physics A* 20(18): L1305-L1309.

Shon, A. P. and R. P. N. Rao (2005). Implementing belief propagation in neural circuits. *Neurocomputing* 65-66: 393-399.

Shu, Y., A. Hasenstaub, et al. (2003). Turning on and off recurrent balanced cortical activity. *Nature* 423(6937): 288-293.

Siegelmann, H. T. and E. D. Sontag (1991). Turing Computability with Neural Nets. *Applied Mathematics Letters* 4(6): 77-80.

Siegelmann, H. T. and E. D. Sontag (1995). On the Computational Power of Neural Nets. *Journal of Computer System Science* 50(1): 132-150.

Silberberg, G., A. Gupta, et al. (2002). Stereotypy in neocortical microcircuits. *Trends in Neurosci.* 25(5): 227-230.

Silver, R. A., J. Lübke, et al. (2003). High-Probability Uniquantal Transmission at Excitatory Synapses in Barrel Cortex. *Science* 302(5652): 1981-1984.

Skoglund, T. S., R. Pascher, et al. (1996). Heterogeneity in the columnar number of neurons in different neocortical areas in the rat. *Neuroscience Letters* 208: 97-100.

Somers, D. C., E. V. Todorov, et al. (1998). A local circuit approach to understanding integration of long-range inputs in primary visual cortex. *Cerebral Cortex* 8(3): 204-217.

Sommer, F. T. and P. Dayan (1998). Bayesian retrieval in associative memories with storage errors. *IEEE Trans. on Neural Networks* 9(4): 705-713.

Sporns, O., G. Tononi, et al. (2000). Theoretical Neuroanatomy: Relating Anatomical and Functional Connectivity in Graphs and Cortical Connection Matrices. *Cerebral Cortex* 10(2): 127-141.

Spratling, M. W. and M. H. Johnson (2002). Preintegration Lateral Inhibition Enhances Unsupervised Learning *Neural Computation* 14(9): 2157-2179.

Steinert, R., M. Rehn, et al. (2006). *Recognition of handwritten digits using sparse codes generated by local feature extraction methods*. European Symposium on Artificial Neural Networks (ESANN'2006), Bruges, Belgium, d-side

Storkey, A. (1997). *Increasing the capacity of the Hopfield network without sacrificing functionality*: 451-456. International Conference on Artificial Neural Networks (ICANN'97), Lausanne, Switzerland, Springer.

Storkey, A. J. and R. Valabregue (1999). The basins of attraction of a new Hopfield learning rule. *Neural Networks* 12(6): 869-876.

Strey, A. (2003). On the suitability of SIMD extensions for neural network simulation. *Microprocessors and Microsystems* 27(7): 341-351.

Stringer, S. M., G. Perry, et al. (2006). Learning invariant object recognition in the visual system with continuous transformations. *Biol. Cybern.* 94: 128-142.

Stringer, S. M. and E. T. Rolls (2000). Position invariant recognition in the visual system with cluttered environments. *Neural Networks* 13: 305-315.

Sultan, F. (2002). Analysis of mammalian brain architecture. *Nature* 415: 133-134.

Sundararajan, N. and P. Saratchandran (1998). *Parallel Architectures for Artificial Neural Networks: Paradigms and Implementations*, Wiley.

Sur, M., A. Angelucci, et al. (1999). Rewiring Cortex: The Role of Patterned Activity in Development and Plasticity of Neocortical Circuits. *Journal of Neurobiol.* 41: 33-43.

Swindale, N. V. (1998). Cortical organization: Modules, polymaps and mosaics. *Current Biology* 8: R270-273.

Szentágothai, J. (1975). The `module-concept' in cerebral cortex architecture. *Brain Res.* 95(2-3): 475-496.

Tanaka, T. and M. Yamada (1993). The Characteristics of the Convergence Time of Associative Neural Networks. *Neural Computation* 5(463-472).

Tang, H., K. C. Tan, et al. (2004). A Columnar Competetive Model for Solving Combinatorial Optimization Problems. *IEEE Trans. on Neural Networks* 15(6): 1568-1573.

Telfeian, A. E. and B. W. Connors (2003). Widely integrative properties of layer 5 pyramidal cells support a role for processing of extralaminar synaptic inputs in rat neocortex. *Neuroscience Letters* 343(2): 121-124.

Thomson, A. M. and A. P. Bannister (2003). Interlaminar Connections in the Neocortex. *Cerebral Cortex* 13(1): 5-14.

Togawa, F., T. Ueda, et al. (1991). *Receptive field neural network with shift tolerant capability for Kanji character recognition*: 1490-1499. International Joint Conference on Neural Networks (IJCNN'91), Singapore, IEEE.

Torkkola, K. (2003). Feature Extraction by Non-Parametric Mutual Information Maximization. *J. Machine Learning Res.* 3(7-8): 1415-1438.

Toulouse, G. (1989). Perspectives on neural network models and their relevance to neurobiology. *Journal of Physics A* 22(12): 1959-1968.

Toya, K., K. Fukushima, et al. (2000). Bistability of mixed states in a neural network storing hierarchical patterns. *J. Phys. A: Math. Gen.* 33: 2725-2737.

Treves, A. (1991). Dilution and sparse coding in threshold-linear nets. *J. Phys. A: Math. Gen.* 24: 327-335.

Treves, A. (2005). Frontal latching networks: a possible neural basis for infinite recursion. *Cognitive Neuropsychology* 22(3-4): 276-291.

Tråvén, H. G. C. (1993). *On Pattern Recognition Applications of Artificial Neural Networks*. Ph. D. Thesis, Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden.

Tseng, S.-T., A. B. Yeh, et al. (2003). A Study of Variable EWMA Controller. *IEEE Trans. on Semiconductor Manufacturing* 16(4): 633-643.

Tsodyks, M. V. and M. V. Feigelman (1988). The Enhanced Storage Capacity in Neural Networks with Low Activity Level. *Europhysics Letters* 6(2): 101-105.

Turner, A. M. and W. T. Greenough (1985). Differential rearing effects on rat visual cortex synapses. I. Synaptic and neuronal density and synapses per neuron. *Brain Res.* 329(1): 195-203.

Türel, Ö., J. H. Lee, et al. (2005). Architectures for nanoelectronic implementation of artificial neural networks: new results. *Neurocomputing* 64: 271-283.

Ueda, N. and R. Nakano (1994). A New Competitive Learning Approach Based on an Equidistortion Principle for Designing Optimal Vector Quantizers. *Neural Networks* 7(8): 1211-1227.

Wahlgren, N. and A. Lansner (2001). Biological evaluation of a Hebbian-Bayesian learning rule. *Neurocomputing* 38-40: 433-438.

Wallis, G. and E. T. Rolls (1997). Invariant Face and Object Recognition in the Visual System. *Progress in Neurobiology* 51: 167-194.

Watanabe, T., K. Kimura, et al. (1993). A single 1.5-V digital chip for a 106 synapse neural network. *IEEE Trans. on Neural Networks* 4(3): 387-393.

Watts, D. J. and S. H. Strogatz (1998). Collective dynamics of 'small-world' networks. *Nature* 393(6684): 440-442.

Vaughan, F. A., Grove, D.A. and Coddington, P.D. (2003). *Communication Performance Issues for Two Cluster Computers* 16: 171-180. Twenty-Sixth Australasian Computer Science Conference (ACSC2003), Adelaide, Australia.

Wen, Q. and D. B. Chklovskii (2005). Segregation of the Brain into Gray and White Matter:a design minimizing conduction delays. *PLoS Comput. Biol.* 1(7): 617-630.

Verleysen, M., B. Sirletti, et al. (1989). A high-storage capacity content-addressable memory and its learning algorithm. *IEEE Trans. on Circuits and Systems* 36(5): 762-766.

Wersing, H., J. J. Steil, et al. (2001). A Competitive-Layer Model for Feature Binding and Sensory Segmentation. *Neural Computation* 13(2): 357-387.

Widrow, B. and M. E. Hoff (1960). *Adaptive switching circuits*: 96-104. IRE WESCON Convention Record, New York.

Williams, R. W. (2000). Mapping genes that modulate brain development: a quantitative genetic approach. *Mouse Brain Development*. A. M. Goffinet and P. Rakic. New York, Springer Verlag**:** 21-49.

Willshaw, D. J., O. P. Buneman, et al. (1969). Non-holographic associative memory. *Nature* 222: 960-962.

Vogt, H. and A. Zippelius (1992). Invariant recognition in Potts glass neural networks. *J. Phys. A: Math. Gen.* 25(8): 2209-2226.

Wong, T. P., P. M. Campbell, et al. (1998). Synaptic numbers across cortical laminae and cognitive performance of the rat during ageing. *Neuroscience* 84(2): 403-412.

Woolsey, T. A. and H. V. d. Loos (1970). The structural organization of layer IV in the somatosensory region (S I) of mouse cerebral cortex *1: The description of a cortical field composed of discrete cytoarchitectonic units. *Brain Res.* 17(2): 205-242.

Wu, J.-M. and S.-J. Chiu (2001). Independent component analysis using Potts models. *IEEE Trans. on Neural Networks* 12(2): 202-211.

Wuagh, F. R. and R. M. Westervelt (1993a). Analog neural networks with local competition. I. Dynamics and stability. *Phys. Rev. E* 47(6): 4524-4536.

Wuagh, F. R. and R. M. Westervelt (1993b). Analog neural networks with local competition. II. Application to associative memory. *Phys. Rev. E* 47(6): 4537-4551.

Yanai, H.-F. and S.-i. Amari (1996). Auto-associative memory with two-stage dynamics of nonmonotonic neurons. *IEEE Trans. on Neural Networks* 7(4): 803-815.

Yang, Y. and G. I. Webb (2002). *A Comparative Study of Discretization Methods for Naive-Bayes Classifiers*: 159-173. Pacific Rim Knowledge Acquisition Workshop (PKAW'02), Tokyo, Japan.

Yedidia, J. S. (1989). Neural networks that use three-state neurons. *Physical Review A* 22(12): 2265-2273.

Yen, G. and A. N. Michel (1992). A learning and forgetting algorithm in associative memories: the eigenstructure method. *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing* 39(4): 212-225.

Young, M. P., J. W. Scannell, et al. (1994). Analysis of Connectivity: Neural Systems in the Cerebral Cortex. *Reviews in the Neurosciences* 5(3): 227-249.

Yousef, T., T. Bonhoeffer, et al. (1999). Orientation topography of layer 4 lateral networks revealed by optical imaging in cat visual cortex (area 18). *European Journal of Neuroscience* 11(12): 4291-4308.

Yuille, Smirnakis, et al. (1995). Bayesian Self-Organization Driven by Prior Probability Distributions. *Neural Computation* 7: 580-593.

Zhang, K. and T. J. Sejnowski (2000). A universal scaling law between gray matter and white matter of cerebral cortex. *Proc. Natl. Acad. Sci. USA* 97(10): 5621-5626.

Zhang, Z.-W. and M. Deschênes (1997). Intracortical Axonal Projections of Lamina VI Cells of the Primary Somatosensory Cortex in the Rat: A Single-Cell Labeling Study. *The Journal of Neuroscience* 17(16): 6365-6379.

Zhu, S. and D. Hammerstrom (2002). Simulation of associative neural networks. *In proceedings of the 9th International Conference on Neural Information Processing, ICONIP '02* 4: 1639-1643.

Zurada, J. M., I. Cloete, et al. (1996). Generalized Hopfield networks for associative memories with multi-valued stable states. *Neurocomputing* 13(2-4): 135-149.