This is to certify that the
dissertation entitled

CORTEX-INSPIRED
GOAL-DIRECTED RECURRENT NETWORKS FOR
DEVELOPMENTAL VISUAL ATTENTION AND
RECOGNITION WITH COMPLEX BACKGROUNDS

presented by

MATTHEW LUCIW

has been accepted towards fulfillment
of the requirements for the

Ph.D. degree in Computer Science

Juyang Weng
Major Professor's Signature

May 14, 2010
Date

Cortex-inspired Goal-directed Recurrent Networks for Developmental Visual Attention and Recognition with Complex Backgrounds

By

Matthew Luciw

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science

2010

Abstract

Cortex-inspired Goal-directed Recurrent Networks for Developmental Visual Attention and Recognition with Complex Backgrounds

By

Matthew Luciw

It is unknown how the brain self-organizes its internal wiring without a holistically-aware central controller. How does the brain develop internal object representations for a massive number of objects? How do such representations enable tightly intertwined attention and recognition in the presence of complex backgrounds? Most vision systems have not included top-down connectivity or treated bottom-up and top-down separately. Yet almost all excitatory pathways in the visual cortex are bidirectional; evidence suggests the top-down connections are not fundamentally different from bottom-up connections. This dissertation presents and analyzes a hierarchical self-organizing type of network with adaptive excitatory bottom-up and top-down connections. This two-way network takes advantage of grounding — both the sensory end (visual patches) and motor end (action control) are input ports. Internally, local neural learning uses only the co-firing between the pre-synaptic and post-synaptic activities. Such a representation automatically boosts action-relevant components in the sensory inputs (e.g., foreground vs. background) by increasing the chance of only action-related feature detectors to win in competition. After learning, the bidirectional networks showed topographic semantic grouping and modular connectivity. It is shown how and why such modular networks can take advantage of recurrent excitation for recognition. In Where-What Network-3, top-down connections enabled type-based and location-based top-down attention and synchronization of neurons over multiple levels to bind features into holistic representations.

Dedication

To my parents.

Acknowledgements

I truly have to thank my advisor, Professor Juyang (John) Weng. John has been a great advisor for me, and I've learned so much from him. He's been an endless source of inspiration and motivation. I'd like to thank professors Charles Ofria, Kelly Mix, and George Stockman for serving on my committee and providing critical comments on this work. A special thanks goes to George Stockman, for being tremendously supportive throughout my time in graduate school and for teaching me about Computer Vision, and to Professor Wayne Dyksen, who has been a great help and an inspiring person to work with as a teaching assistant. I've been influenced by many professors through taking their courses at MSU. In particular, classes taught by Barbara Abbott, Betty Cheng, Richard Enbody, Abdol Esfahanian, John Forsyth, Anil Jain, Devin McAuley, Mark McCullen, Bill Punch, Charles Ofria, Charles Owen, Richard Reid, George Stockman, Mark Sullivan, Juyang Weng, and Anthony Wojcik were influential and inspiring. The Computer Science graduate director, professor Eric Torng, was extremely supportive.

This dissertation represents the completion of my Ph.D. at Michigan State. There are many people that I'd like to thank. Some past and present members of the Embodied Intelligence (EI) Lab at MSU provided helpful discussions, assistance, and collaborations. A great thanks goes out to collaborators Zhengping Ji, Laura Grabowski, Shuqing Zeng, and Hanqing Zhao. I'd like to especially thank EI Lab members Paul Cornwell and Mojtaba Solgi for their discussions and support. Chuck Bardel gets a special mention. Others at MSU I'd like to mention for their assistance in various ways are Zubin Abraham, Alexis Ball, Tyler Baldwin, Matt Gerber, Faliang Chang, Xiao Huang, Xiaoan (Dustin) Li, Mayur Mudigonda, Brian McCulloch, Kajal Miyan, Vikram Melapudi, David Mulder, Zahar Prasov, Yan Wang, Liu Yang, and Nan Zhang. Finally, I'd like to thank my sister Julia, my Dad, and my Mom for their encouragement and support throughout my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

This dissertation is concerned with recurrent networks for attention and recognition in developmental agents. It focuses on vision, but it is applicable to other modalities. The work here is on *general-purpose* attention and recognition. In this sense, the methods are not meant to directly apply to a single particular engineering problem, but are instead motivated by the issue of how to build a machine that could understand the visual world. Such machines could be taught to solve specific problems, but it is not with a particular problem in mind that they are designed.

It seems logical to take a reductionist view of the mind. In this light, the mind, mental states, attitudes etc. are all encapsulated in the brain. They are all explainable by brain operation. Arguments about the existence of intuitive phenomena like mental states or qualia are attributable to confusion about what these terms are actually referring to. Such terms like "mental state", "thinking", or "feel" fundamentally seem to reflect misunderstanding of what is actually happening in the brain. The introspective style of reasoning about the mind has been very useful. It may be possible to formulate some higher-level theory of mind that explains intelligent behavior and can be used in building intelligent machines. But results from those working admirably at this scale over many years have illuminated some difficulties of this approach.

As discussed elsewhere by Weng, symbolically programmed AI's have been brittle [117]. Attempts to model intelligent behavior suffered from such brittleness problems since it seems impossible for the programmer to predict all the rules of all the cases the machine should be able to handle. To deal with this, researchers turned their attention to methods of learning, inspired by the thought that if the machine learns, then providing it with enough experience should be all that is needed. Learning methods have led to very promising results and the thriving field of machine learning. But learning methods have mostly focused on solving single problems, thus generating artificial intelligences that can perform single tasks. Progressing towards AIs that can learn multiple tasks, without the programmers or modelers knowing the tasks in advance, is difficult with many learning methods that work very well for a single task. For example, many learning methods suffer from the well-known long-term memory problem: since they are designed to utilize all their resources to greedily learn any task, they will forget already learned tasks. There is also the related scaffolding problem: most learning methods that have learned one task have not made it any easier for the machine to learn a different but related task. Yet we can use our past experience to make it easy for us to learn some new skills in only a few tries, or even in one shot. The developmental approach, also known as autonomous mental development (AMD) [123], has emerged and aims to take these issues into account. The goal of AMD is to build a developmental program so the machine can learn skills from experience in a way that does not suffer from the long-term memory problem or the scaffolding problem. The focus has shifted towards the study of general purpose learning mechanisms.

Over the history of AI, many relatively successful approaches were formulated as inspired by another field or combination of fields, such as psychology, biology, chemistry, or neuroscience. It is an open question as to which scale is most appropriate for modeling a developmental learning program. Recently, technology has advanced to a point that the amount of data coming out of the neuroscience field has exploded, and much is now known about the visual pathway, and about some mechanisms of computation in cortex. Much is still unknown, however. Theories and results from psychology have been incredibly influential and illuminating in understanding visual attention and recognition. Connectionist approaches have been inspired by both psychology and neuroscience. Background over multiple related fields will be discussed further in Chapter 2.

Putting together some key knowledge about the brain may be leading towards solving the puzzle of how a successful developmental program could be built. Cortical neurons can be considered as "internal representation" as demonstrated by stimulus-response selectivity. Neurons represent sensory stimuli and also movements (shown via firing-movement correlations). Studies in mapping the brain's function has shown that different areas in each animal are reliably correlated with certain function, yet in no way are all the functions of the different areas known. But there is recent evidence that what the neurons represent is not dependent on its physical location, but instead on the area's input [96]. This points to, not a genetically-encoded representative scheme based on brain location, but a shared mechanism of learning that occurs in each of the areas, no matter what location. In this case, different functions emerge since there are different inputs to each area. What is known about neuronal learning is based on synaptic strength modification and so far follows Hebb's principle and, at a lower temporal scale, the principles of spike-timing dependent plasticity (STDP). Some principles may also be emerging based in the neuromodulation of learning. In Chapter 3, I'll discuss a mechanism of general purpose learning, for a layer of neurons[1] inspired by these results. This was called lobe component analysis [122,124] (LCA). LCA satisfies many of the constraints of a developmental program

[1] I do not claim that these artificial neurons are functionally isomorphic to actual neurons. They represent units of information integration and computation, inspired by actual neurons or neuron groups.

appropriate for AMD.

The circuitry of the mature cortex is very complicated. Again the notion of input-driven self-organization is encouraging. Over multiple layers, the neuronal learning method discussed in Chapter 3 leads to selective wiring — automatic organization of the network connectivity. Input-driven selective wiring over multiple layers is the focus of Chapter 4, which focuses on the purpose and effects of the top-down connections. These connections cause recurrence in the networks; the chapter presents a method and its analysis for utilizing top-down connections for a biased compression of information, leading to internal representations that are more useful for successful behavior given a limited resource. The top-down connections do not represent supervised learning in the sense of gradient descent methods — they operate the same as bottom-up connections. Top-down and lateral excitation together in the learning phase produces modular networks that show topographic class grouping — layers with representation spatially organized so that neurons that represent the same action but different sensations are grouped together. It is so far unknown how cortical areas could emerge that represent very different (in a physical sense) stimuli, like the parahippocampal place area. Methods, analysis and results about running the mature multilayer networks as recurrent dynamical systems is presented in Chapter 5. The modularity of a network with topographic class grouping leads to helpful recurrent effects that networks without modularity do not have.

Mechanisms and understanding of top-down connections, from chapters 4 and 5, are integrated into the work in Chapter 6. It focuses on recurrent networks for general purpose visual attention and recognition called Where-What networks (WWN) [47,48,63]. WWNs treat attention and recognition in a unified way, as it seems the brain does not deal with these two problems independently. This chapter discusses the architecture, training, and operation of WWNs for bottom-up and top-down attention of any foreground over complex backgrounds. When there are multiple objects in the visual scene, the binding problem becomes an issue. In WWN, synchronization via recurrence and information flow control lets the networks deal with the binding problem without using combination neurons.

In, Chapter 7, I summarize the contributions and results in the dissertation, and a few possible future directions are briefly explored.

Chapter 2

Background

The topic material of this dissertation is at an intersection of computer vision, pattern recognition (models of learning), computational neuroscience, artificial intelligence, and psychology (studies of visual representation and attention). This chapter discusses relevant background information from these fields. The purpose of this chapter is to introduce the problems that motivated this work and to provide context for the later chapters.

2.1 Visual Representation

One can view visual perception as a problem that our visual systems must solve. Study and analysis of this problem has uncovered the amazing capabilities of biological vision. A few will be discussed here. First, our ability to recognize an object that we know seems extremely robust over different viewing distances and angles. Second, we seem to have this capability for a huge number of objects. Underlying these two capabilities are questions of visual representation. Visual representation is internal storage and mechanisms for interpreting and explaining visual stimuli. What representations are appropriate for human-level capabilities in object vision?

2.1.1 Viewpoint Invariance

Our ability to recognize objects is very robust. We can reliability recognize single objects that we know over many possible variations. Consider the following simplified setting for a single object: a viewer is some fixed distance from the object, which is within the viewer's field of view. The object exists in 3D world space but is represented as a pattern of intensities or colors on the viewer's retinal image, which we can consider as 2D. Let's assume the object completely fits within the retinal image, is central on it, and there is no background. The object has its own internal 3D coordinate frame, which we'll assume has its origin at the center of the object. "Complete" *viewpoint invariance* implies that if the viewer is able to identify (ID) the object for any one rotation, the viewer can ID it over all possible rotations. A softer definition allows the viewer to see several different views first or constraints the testing set. Note the viewer doesn't see the object temporally rotate. Whether we have such a capability has been the subject of debate as summarized by Edelman ([30], p. 178), due to seemingly conflicting data.

If the possible rotations are restricted occurs so the object's appearance rotates in an affine way on the retinal image, then all the information needed for recognition is contained in a single view. Could every point on the rotated image be mapped to another point on an internal 2D stored view — a canonical template? Recognition could be accomplished by "mentally" rotating the template until it matches the object in the image. But consider the more realistic case where the rotations may be over all three dimensions. There is no longer a mapping of from a 2D template of one view to the visual perception, showing another view. If we extend the 2D template matching concept of storage, then each object we've learned would have a massive number of views associated with it, some of which can be very different. Think of looking straight down into a cup compared to viewing it from the side. It does not seem feasible nor optimal to store 2D template representations of every view of every object that we can recognize.

Use of “3D templates” was the subject of much investigation. An internal model of the world-based 3D shape would seem to work well for any particular object. An experiment by Shephard and Metzlar, 1971 [90] supports the idea that objects are stored via an internal 3D representation. They tested subjects’ reaction time in a same/different mental rotation task using 3D objects, similar to tetris blocks. Two views of an object were presented, which showed either the same object rotated along the vertical axis, or two mirror images of the object rotated along the vertical axis. The reaction time results showed a linear dependency on rotation angle, which suggests we can mentally rotate 3D internal representations. Intuitively, when given a view from a novel, but somehow familiar 3D object, we do not have much trouble imagining how it would look from another angle. The idea of using 3D templates is object-based, and faces the problem of reconstruction from views. David Marr [70] provided an fundamental theory of vision leading to 3D reconstruction, and the more recent technique of SLAM [28] can reconstruct surfaces from views. SLAM has shown to work well for navigation.

There was quite a bit of debate about whether internal representation is (or should be) object-based or view-based. There is support for view-based representation from experiments showing poor performance of humans on viewpoint invariance for certain objects, such as complex bent ‘paperclip’ objects [87]. Some objects (e.g., faces) are more similar over multiple views than others (e.g., paperclips). A core idea is that some of an object’s parts change less over a set of views, in which the entire view of the object changes significantly. A view-based system with viewpoint invariance over about 180 degrees were built for faces and cars [88], via decomposition of the image into parts.

2.1.2 Object Representation

Should objects be represented holistically or broken up into parts?

David Marr ([70], 1982) explained how holistic 3D-like templates could be learned from a viewer's perspective. Inspired by neuroscience (he was one of the first to advocate both a functional and computational study of the brain), Marr proposed that the visual recognition has three stages. First was the primal sketch, a map of feature activation over the scene (e.g., edge locations were provided by edge detection). The 2.5D sketch incorporates grouping, texture, and some depth information (depth information is available since two eyes give two different perspective views, which can be used to infer disparity and surface — local 3D shape — information). The third stage is the manipulation of the 2.5D sketches towards a 3D reconstruction. Marr's theory has been significant in the history of vision research. But many results about the top-down nature of visual processing have not supported a geometric reconstruction. When one considers a "visual task", such a data-rich and precise representation is probably not necessary. In particular it ignores what the agent's purpose, or goal, is. The theory advocates that all shape information is important. How much of the surface and shape information that is useful depends on the potential use the viewer can make of it. The challenge is how to represent all the available information so that the behavior becomes successful.

As discussed above, parts of an object over views can change little while the entire object changes a lot over the same views. Viewpoint invariance could be accomplished through such local invariants. This was the inspiration behind Biederman's "Recognition by Components" (RBC) [9]. RBC object representations are composed of volumetric building blocks called geons — a basic primitive 3D shape, such as a cylinder — along with inter-geon relationships. The geons themselves are stored internally and must be recognized with the same effort over any rotation (unless they are occluded). He gave three conditions [10] to enable viewpoint invariant recognition via RBC. If the geon structural description (GSD) is unique for an object, and possible views are not nondistinctive, then viewpoint invariance can occur after seeing one view. Object recognition in this model boils down to comparing the GSD of the current view and comparing it with GSDs for known objects. When he added a single geon to a complex paperclip-like object, the response time of subjects to different views decreased dramatically. Tarr [99] claimed that this result can be explained as the viewer is simply looking for the single added easy-to-detect feature. He replicated this experiment, but added more geons (three and five). He showed that as the number of geons added increased, the viewpoint invariance ability of the subjects decreased. Edelman, Tarr and others helped move the field towards viewer-based methods of representation and recognition.

The view-based models led to appearance-based methods in Computer Vision — representation and recognition from sets of digital images. An initial problem for this idea was that it seemed like a model would have to be automatically built from data to explain each pixel of an image, and this dimension is very large. One way to avoid the dimensionality issue is to map a set of images onto a lower-dimensional manifold. Turk and Pentland showed that, via principal component analysis (PCA) [113], a set of image views could be linearly mapped to a lower-dimensional space, and the new dimension depended on the complexity of the variations in the data. Due to theory of PCA, the lower dimensional eigenspace is optimally representative in a linear sense. Murase and Nayar [76] showed how to generate and parameterize complete eigenspaces for an object. By complete, it means the eigenspace contains all possible views, even those not seen. However, the method was rather computationally expensive to update as new training views were added. Additionally it could not tolerate occlusions.

2.1.3 Local Appearance Hierarchies

The occlusion problem with global appearance methods led to local appearance methods where smaller image windows were used as the inputs of the algorithms [24]. Interestingly, higher-order principal components from smaller windows over multiple objects resembled the features detected by some known early-pathway visual neurons, while lower order principal components did not seem to show useful structure. These local features seemed similar no matter which objects were used in the data. Also, no matter what the objects used, the resulting features were nearly the same. Since PCA is optimal, this suggests that the local statistics over all views is not class dependent. We can assume evolution led to brains that can take advantage of these local statistics. As a simulated model of V1 orientation selectivity, a basis of Gabor filters [79] fits some V1 neurons fairly well. Could internal representation for local appearance be handled through such a basis?

Even though such modeled filters (e.g., parameterized Gabor filters) can fit V1 orientation selectivity well, one basis (or layer) of local filters alone cannot explain object representation and recognition. An efficient coding of many-objects representation from parts requires a hierarchy of features. Feature hierarchies seem to be important for how the brain solves the problem of many-objects representation [86]. It is not known whether such a structure is a necessary condition for many-objects representation, however. In hierarchical systems, the receptive field (area of sensor detected by the filter) and complexity of features increases towards higher layers. Fukishima (Neocognitron [36], 1983), Weng (Cresceptron [118, 119], 1992), and LeCun (LeNet [58], 1998) implemented notable vision systems using a local-to-global hierarchical approach. A hierarchical representation allows compositionality — shared parts between objects — and thus an efficient coding of many possible objects. Another advantage of hierarchies is a robust tolerance to many variations in object deviation due to slight tolerance to deviations of local features.

Thus, invariance should emerge in a local-to-global way, but the exact mechanisms for this to happen remain controversial. “Max pooling”, in which identical filters in different nearby locations become represented by the single filter with the maximum firing in the group [86] seems to aid networks in achieving invariance. In almost all implementations, feature hierarchies utilize only feedforward activation. In mature cortex, speed of processing of object detection indicates that feedforward activity is probably sufficient for recognition [103]. However, another result indicates that feedback causes better performance [104]. There may be other roles for feedback, such as in learning the features in the first place.

Local feature hierarchies have not yet found the best method to set the features on each layer. The evidence of input-driven functional emergence in the brain [56] suggests a tantalizing prospect: perhaps the same learning mechanism is active in groups of neurons in all the different areas of the visual pathway. Appropriate diversity of function could emerge given this learning rule and appropriate input. If this learning rule can locally compress information efficiently and effectively, the global problem of setting features at all hierarchical levels could be solved by applying this same method to each of the levels concurrently.

To test a learning algorithm's suitability for such a method, we can see if it can extract features similar to those in V1 from the same type of data V1 may be interacting with. Thus, any learning rule that develops orientation filters from *natural images* can be considered a candidate in general. But only an in-place [124] — also called local learning — mechanism is also biologically plausible. Weng proposed Lobe Component Analysis (LCA) as a candidate for this learning rule, which is nearly in-place. Since LCA develops orientation selective neurons from natural input, it passes as a possible candidate. We showed the advantages of LCA over other Hebbian-learning rules in [122], as summarized in Chapter 3. Chapter 4 describes how LCA can be included in a general framework with bottom-up, lateral, and top-down connections. It's shown how, when using top-down connections, an efficient compression (which prioritizes relevant information) emerges.

2.2 Visual Attention

Recognition operates together with attention to transform an image into a meaningful image. A fundamental problem of recognition is the segmentation problem. Which part of the scene is the foreground, to be recognized? The rest of the scene is considered the background. Realistic backgrounds can have very complex visual structure and may or may not show any other objects. This problem reflects the chicken-egg nature of segmentation (attention) and recognition: it seemingly can't be determined if the edges, colors, etc. belong to the object (figure) or the background until the object is recognized, yet some grouping must occur before attention knows what to select. We also have some explicit internal control over what the foreground is. Consider the famous image in which we can see either two faces or a vase as the foreground (but not both), depending on what one tries to see.

2.2.1 Selection

Selective attention refers to some mechanisms by which an agent recodes its sensory information into a simpler, more useful form. Simplified relevant information is necessary for cognitive processes, such as decision making. Attention is essential for artificial agents that learn intelligent behavior in complex unconstrained environments, especially those that utilize vision. Attention is called selective, in that a subset of the sensed information is suppressed. What it means for information to be suppressed is not explicitly known. It may not be simply removed from being processed. Even in situations when a person cannot remember something sensed, that information can still influence that person's actions [67].

Selective attention is separated into bottom-up selection and top-down selection [27]. Bottom-up selection is not explicitly controlled: salient foregrounds will automatically "pop out" at the viewer. Sometimes, there is not a single salient object in the image, but several objects, and the most salient is what is attended first. Top-down attention, a fundamental part of visual attention [27], is selective for important locations or features biased by goal or task of the agent. Given the same scene with the same eye fixation, but two different top-down biases, the representation of the information that reaches the later stage can be very different. For example, imagine the differences between what a vehicle's driver tends to attend to compared to a passenger, even if they look in the same direction.

Treisman's Feature Integration Theory (FIT) [109] has been an extremely influential model of representation and attention, which described selection based on saliency. In FIT, objects are decomposed into features, which themselves can be recognized without attention, but a conjunction of features requires an attentional spotlight to "shine" on that location, which selects that location. FIT introduced the idea of separate feature maps concerned with different dimensions of stimuli (i.e., color, orientation, direction of movement, and disparity) feeding into a single *master map* of salient locations. Koch and Ullman [53] proposed a computational saliency model, implemented later by Itti and Koch [46], in which winner-take-all operation of neurons on the master map led to both binding and attention location.

Saliency methods have been coupled with recognition. An example is NAVIS (Neural Active Vision) by Backer *et al.* [2]. An issue with feature hierarchies is the corruption of the original information [111]. Olshausen, Anderson and Van Essen proposed a neural computing model of dynamic information routing to go along with a saliency map structure [78] so that the original information in the attended area could be sent to a recognition network. Each object is stored inside the network as an object-based reference frame, which is used for recognition via an associative memory similar to those demonstrated by Hopfield [41]. Control neurons set "shifter circuits" over multiple levels, which will normalize a part of the image in scale and location for comparison with the object-based reference frame.

Non-biologically-inspired methods for top-down attention include the Viola and Jones approach [115], which is designed to find a particular class, such as faces. The histogram of gradients approach, used for pedestrian detection [22], uses local histograms of image gradient orientations as features to predict a certain classes presence well. In both of these, the goal of attention selection is determined beforehand.

Other approaches have combined bottom-up and top-down attention. Desimone and Duncan [27] claimed that top-down selection occurs primarily by gating the different channels of object information; in terms of feature-based attention it may be based on comparison with a feature template stored in short term memory. CODAM [100] models attention as a control system, and uses a goal module to process bottom-up signals and hold internally generated goals, generating goal-signals that bias lower-level competitive processing. The method in [75] modifies gains and selects scale of processing to produces different saliency maps, specific for certain classes.

A few researchers have proposed connectionist models for selecting and matching an object in the image to a canonical internal reference frame. Examples include Olshausen, Anderson and Van Essen [78] and Tsostos [111]. Deco and Rolls, 2004 [25], created a biologically inspired network for attention and recognition where top-down connections controlled part of attention, but were not enabled in the training phase due to instabilities they caused. Where-What networks are a biologically plausible developmental model that integrates both bottom-up and top-down modes of attention and recognition, without being limited to a specific task [47, 48, 63]. WWN does not use a master map or internally stored canonical objects [48, 63].

2.2.2 Binding Problem

The binding problem is a fundamental problem of attention, and one that must be solved for local feature hierarchies. Once the scene has become represented by separate feature activations, how can a subset of those activations be recognized as an object? How could a network select the features that actually belong to the object and not select features that don't?

Much evidence does not support the idea that we represent objects in a holistic representation. Experimental results showed the existence of illusory conjunctions [51, 108], which suggests there are separable features in object representation. Illusory conjunctions occur when there are fast visual changes, and feature dimensions of the actual observed items are mixed up upon being reported by the viewer. For example, if one is presented with a red B and a green S very quickly in sequence, upon reporting it back one may say there was a red S. The local-to-global feature hierarchies discussed above also represent the scene in a disintegrated way. Separable features provide representation efficiency, but must eventually be re-integrated for understanding of any meaningful thing in an image. The binding problem concerns the mechanisms of information integration into understandable wholes [107]. For example, if we analyze an image based on location of objects and the types of objects separately, how do we ensure any single solution for location and type is in agreement? If a single image contains two different types in two different locations, and the network (correctly) outputs both positions and both types, it is not clear which type corresponds to which position.

FIT offered an theory of binding and an explanation for illusory conjunctions. The most famous supporting evidence for FIT is in feature-based search. Subjects were instructed to find an item among distractors, and this item differed from all the rest by a single simple feature, such as color. In this case, they were able to find it at the same speed (it "pops out") no matter how many distractors there were.

This result suggests simple features are processed throughout the scene in parallel. However, when the object shared each of its features with some distractor, search time increased linearly as a function of the number of distractors, suggesting this conjunction-type search occurs serially. Treisman proposed that objects are internally represented as an object file containing feature information and information about feature relationships. Then, recognition cannot occur unless the file is applied to an actual location on the image. Attention can be placed at only a single location at any time. The spotlight binds features in the same location into an object file, which is compared with stored object files for recognition. Illusory conjunctions would be a result of the spotlight not in a location long enough for binding to occur, leaving a feature "free-floating".

One proposed high-level solution to the binding problem is by *combination neurons* (or neuron assemblies) that represent the entirety of the combination of features for an object, and exist somewhere in the brain. Then each feature's detection is a necessary condition for the combination neuron to fire. In Olshausen *et al.*'s work mentioned above, the associative memory network is in the spirit of the combination neuron approach — somewhere in the network is stored a model of each "whole" that can be recognized.

There are problems with the combination neuron approach, as underlined in [116]. Returning to the position/type example, we could implement another winner-take-all layer of position/type combination neurons, having afferent input from the separated feature layers. But note this scheme runs into the combinatorial explosion problem that we have been trying to avoid by using a hierarchical architecture of separable features in the first place! It is uncertain how all possible combination neurons could be learned without experiencing all combinations. And combination neurons do not support generalization. For example, if a network can recognize both *red car* and *blue hat*, it should be able to also recognize *blue car* even if it has never seen one before. How is a network that sees a set of parts in some combinations able to generalize across other combinations that have not been seen? Thus, for a general-purpose vision system, any network using the combination neuron approach will eventually run into unexpected ambiguities. However, synchrony of firing over multiple levels can alleviate the binding problem without explicit combination cells [116].

Much of the work has investigated temporal synchrony (feature activations correlate in time at some scale). In this work, synchrony results from bidirectional connectivity: bottom-up and top-down connections. In the position/type network, given an image with two objects a and b, let the position detector layer, through feedforward activity and winner-take-all, select position a, but the type detector layer selects type b. The network now has two different pieces of information, both of which are correct, but possibly not synchronized. It's not necessary try all possible combinations; instead it can set one piece and use it to find the other [1]. Following, Treisman's idea of spotlight, neurons in the position map project back to bias neurons at the appropriate positions in the earlier layer. Type-sensitive neurons related to both type a and b are biased in that position, however, only object a is actually in that position. Then, the biased feedforward activation to the type layer leads to type a being selected. This was implemented in our Where-What networks [63] discussed in Chapter 6. Binding through location selection is often effective, since often location information is enough. But it can't handle transparency or occlusion. For these cases, binding within the selected location is also needed, which can be realized by top-down connections within a feature-hierarchy, so that, for example, higher-layer form can bias the related lower-layer edges.

[1] For intuition, consider a quadratic equation in two variables x and y, which has two possible solutions: $\{(a,b),(c,d)\}$. If we know e.g., $x = a$ and $y = d$, we don't have to plug in all four combinations to find a single solution. Instead, we can set $x = a$ and solve for y.

The crucial importance of top-down in unguided attention was realized by Tsotsos. He proved that selecting features that represent an object well based on the activation of the separate feature maps has exponential complexity (is NP-complete [112]). However, guided by knowledge of the object, the search becomes linear in the size of the image. One interpretation of that result is that if we know how the features activate when the object is in each of all possible positions, we can search through all possible positions (linear-time scan). In other words, without some guidance, attention is intractable. Tsotsos' selective tuning model [111] is a multilayer pyramid-like network that uses a complete feedforward pass to find the best candidate location and type and a complete feedback pass to focus attention, by inhibiting non-selected features and locations. Gating units perform selection from the top-down. Some differences between the selective tuning (ST) model and the Where-What networks are that ST uses top-down selection (gating) and top-down inhibition through winner-take-all, while WWN uses top-down excitation through weighted connections; additionally WWN uses multiple motor areas, for controlling and sensing an agent's actions, while ST uses a single area of interpretive output nodes. With respect to the binding problem, by enforcing WTA on the output layer and only using top-down originating from the output layer, ST effectively uses combination neurons, which run into the combinatorial problem.

2.3 Functional Architecture of the Visual Cortex

It has so far been very difficult to map the mind onto the brain. Yet, some evidence and principles are emerging that are very promising. Studies of cortex can lead to intuition in how to design and implement networks.

2.3.1 Distributed Hierarchical Processing

One can think of the cerebral cortex as a vertical hierarchy. In such a hierarchy, sensations from outside stimuli enter at the bottom (such as light interacting with the photoreceptors of the retina) and influences the system, traveling upwards towards motor areas, which control movement and behavior. A study and model of hierarchical organization of the areas of the primate visual system can be found in [71], and later [32] (see [14] for an overview). Each area in cortex exhibits the same type of six-layered organization. Figure 2.1 illustrates this idea by displaying a partial hierarchy of visual areas.

The visual pathway is composed of two separate pathways. Experiments by Mishkin [74] showed that the lower pathway, called the ventral pathway and leading to IT, specializes in "what" information (recognition). The other path is called the dorsal, or "where" pathway. It is implicated in visuomotor reaching, thus encoding object location. Why should there be two different pathways for what and where? For a particular object, the location in the visual field has very little to do with the identity — where the object is seen does not raise or lower the likelihood of its category too much. So, object class is, at least somewhat, invariant to visual field location. And the opposite seems true as well: object location is invariant to its identity. The two diverging pathways come together later at the prefrontal areas, where information is integrated from multiple modalities. The prefrontal areas project to motor areas, which control movement and behaviors like speech. Figure 2.2 shows some of the sensorimotor pathways through human cortex. For each pathway, information flows in both directions.

Neurons fire all along the visual pathway in object recognition. Objects seem to be represented in a hierarchical and distributed way along this pathway. At lower levels (V1 and V2), neurons' firing is selective for features such as oriented edges [42], disparity [6,21], color, and motion [92]. V4 seems to be selective for local

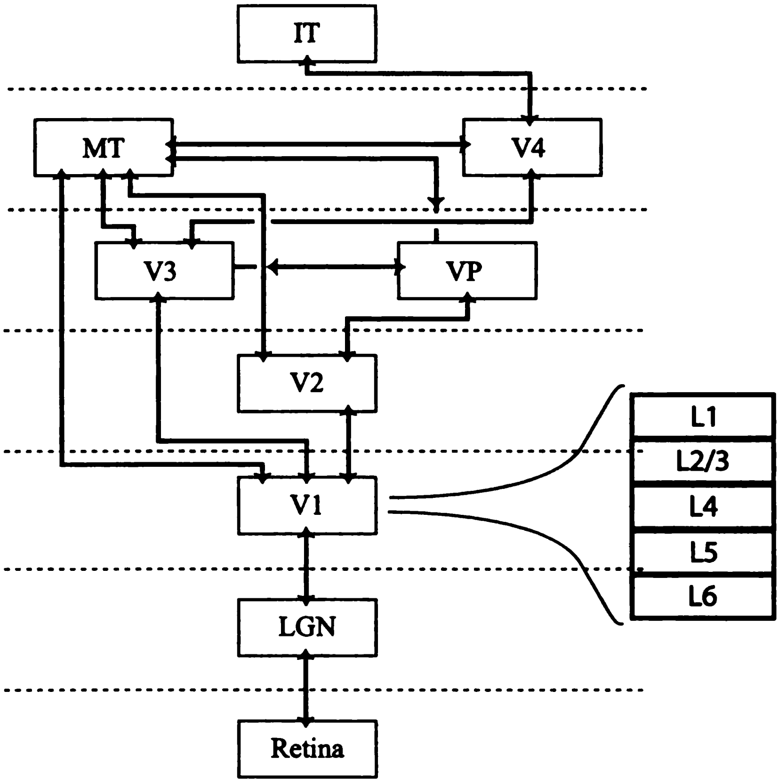


Figure 2.1: As the first figure in this dissertation, I must note the following: Images in this dissertation are presented in color. This figure shows a partial model of a visual hierarchy, starting from sensor cells in the retina and progressing to inferotemporal cortex (IT), an area implicated in visual object recognition. Information progresses through these pathways and beyond and eventually to later motor areas (not shown). All connections shown are bidirectional. Each area is organized into six-layers of highly interconnected neurons. The different areas in cortex, all have this same general six-layer connectivity pattern [50]. A core idea is the hierarchical arrangement of neural areas on many levels, from sensors to motors, with each area's suborganization being six-layered. Figure adapted and modified from [71].

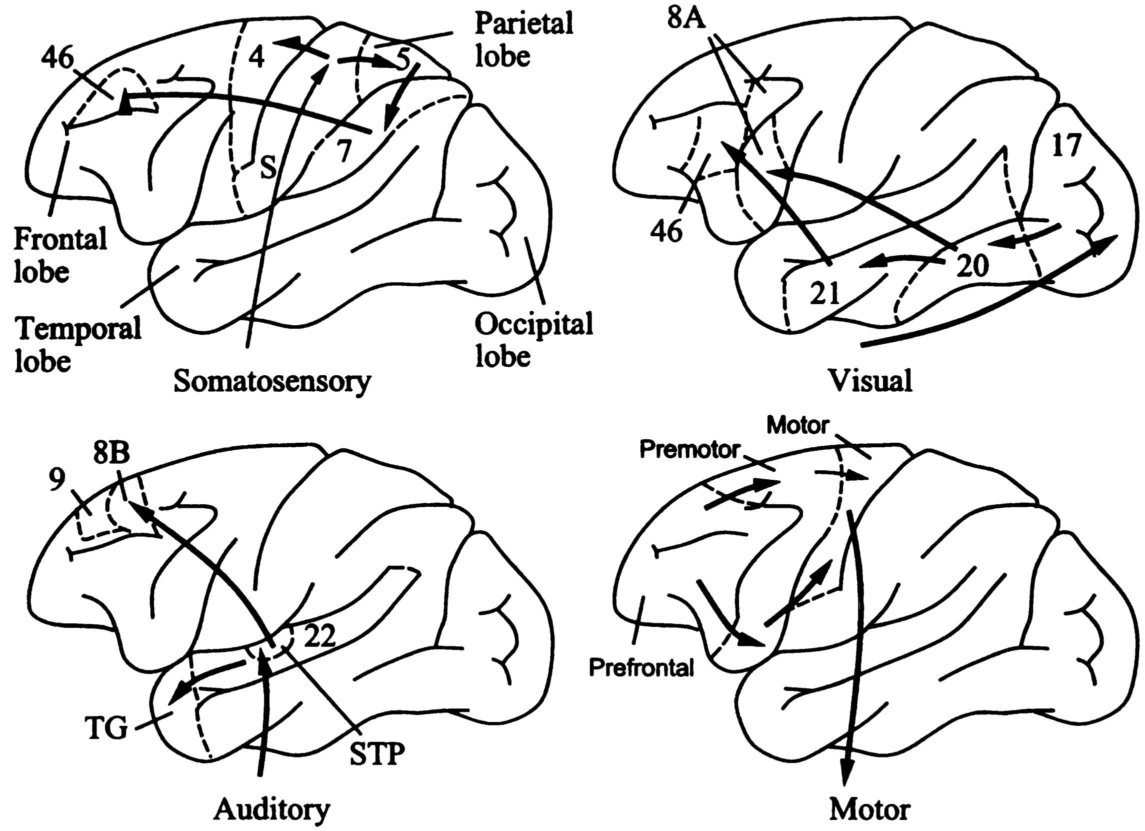


Figure 2.2: Some pathways through different cortical areas for different sensory modalities. The numbers indicate the numerical classification of each cortical area originally by Brodmann in 1909 [50]. There are two different pathways by which visual information travels — it has been found that one pathway is for identity and categorization ("what") and the other has more to do with location (e.g., how to reach for the object - "where"). The somatosensory, auditory and visual information progresses through different pathways, converging at premotor and prefrontal areas. The premotor area goes to the motor cortex, which handles movement and action representation. All paths are bidirectional. Figure courtesy of Juyang Weng.

shape features [83]. In the later IT area, neurons are found to fire in response to abstract things like faces or places (interestingly such neurons are grouped together in these areas). Hubel and Wiesel's discovery of orientation selectivity of V1 neurons inspired the study of "functional architecture" of the visual cortex, which has led to an immense research effort.

Yet mapping the functional architecture is difficult. In 1991, Felleman and Van Essen [33] provided a hierarchical diagram of different cortical areas in vision; additionally they mapped out the connectivity between the visual cortical areas (in the macaque monkey). They organized 32 different visual areas into 14 levels of cortical processing, and many areas were connected (there is a higher probability closer areas are connected), almost always in a bidirectional way. The complexity of the architecture is not encouraging. The functional — e.g., the type of information that is selected for — purpose of only a subset of the neurons in only a few of these 32 areas is known. It's not likely this mapping could be used an explicit blueprint for modeling due to the lack of information about function and its high complexity.

But directly modeling mature structure is probably not necessary, due to evidence supporting a developmental approach. There are general principles of organization behind the complex architecture and multi-area selective wiring.

2.3.2 Input Driven Organization

The concept of a developmental program in an artificial agent is analogous to the genome in a biological agent. In biological agents, development starts within a single cell, called the zygote, containing the genome of the organism. All physical and mental development thereafter is dependent on the genome's coding, meaning that the emergence of behaviors and skills is a process driven by the genes. The human genome contains less than 30,000 genes [95]. Compare that to the estimated 10^{11} neurons and 10^{14} synapses in the central nervous system of a mature adult.

The discrepancy in number between the two estimates implies that all the different cortical functions, such as edge detection, shape detection, face detection, object recognition, motor control, etc. cannot be described in the genome.

Many neurons in the first layer of the visual cortex (known as V1) develop sensitivity to edges at preferred orientations [43]. Are these low-level feature detectors in human vision pre-defined — hardcoded within the genome? Evidence suggests the answer is no. Blakemore and Cooper's 1970 experiment [11] and Sur's work on "rewiring" a ferret's auditory cortex using visual information showed that feature detectors are developed. In [56], an experiment was done on a newborn ferret where input from the visual sensors was redirected to the auditory cortex, and input from the auditory sensors was disconnected. It was found, after some experience, that what would have been the auditory cortex in a normal ferret had altered its representations to function as the visual cortex for the "rewired" ferret. Thus, it is thought that the function of the different cortical areas develops as a result of the same mechanism throughout. The evidence suggests that cortical function and organization is *input-driven*. Therefore the only reason developed neurons will respond to different stimuli is because they adapt to different types of input. That is, the operations done by each neuron are the same, but the input is different, so they will adapt to detect different features. This suggests the same learning mechanisms are active at different areas of the cortex. This notion inspired local learning rules for self-organization — feature extraction and automatic selective wiring — described in Chapters 3 and 4.

2.3.3 Cortical Circuitry

Cortical neurons in any layer derive their representations via input connections from other neurons from three locations: from earlier layers and areas (ascending or bottom-up), from the same layer (lateral), and from later layers and areas (descending or top-down). There are both excitatory and inhibitory connections. About 85% of the connections are excitatory [29].

Excitatory top-down — feedback — connections are more numerous and generally more diffuse than the bottom-up connections. They have been assumed to play a modulatory role, while the bottom-up connections were considered as the directed information carriers [14]. Driving connections move information while modulatory connections modify the activity of the driving connections. But knowledge of the specific computational roles — especially in development — played by the top-down excitatory connections is not yet known. There is evidence that these connections' are quite important for visual classification ability [104]. It is known they play a crucial role in perception over time, which seems to require some recurrence in the circuits. There is much evidence of the top-down connections' impact on visual awareness for functions such as enabling segmentation between an object and its background [102], perceptual filling in [57], and awareness of visual motion [82].

The role of the top-down connections in brain area organization and functional development is very much unknown. In mammalian cortex, later visual cortical areas include functionally specific regions in which neurons that respond to many class variations are spatially localized. Neurons in the inferotemporal (IT) area, along the ventral pathway, have been implicated in object and class recognition [98]. Further, there are areas that seem to have been developed to represent a specific category of stimuli. These stimuli in IT include faces [26], localized within the fusiform face area (FFA) and places, within the parahippocampal place area (PPA). Stimuli such as places do not typically have much physical similarity. Could top-down and lateral excitatory connections lead to these abstractly grouped areas of cortex? The theory and results in Chapter 4 describes a method where top-down connections can cause biased compression of information so that important (relevant) information has a higher priority than irrelevant information; additionally top-down and lateral excitation is shown to lead to such grouped areas. Chapter 5 describes a method in which top-down connections can cause temporal processing.

2.3.4 Smoothness and Modularity

Many cortices, such as the somatosensory, motor, and visual, have been observed to be organized topographically. The smooth topographic organization of orientation selectivity in neurons in primary visual cortex (V1) is classically well known. At higher levels, this smooth organization gives way to a modular organization [15]. How does the cortex develop local topography, which seemingly requires smoothness, yet develop two adjacent areas as modular, which requires a partition that seemingly violates principles of smooth topography? Tootell [105] measured the responses of each of the neighboring FFA and PPA areas (in humans using fMRI) for stimuli of different morphs between faces and houses. They found that the two areas could be considered functionally different modules, since the peaks of the averaged morphed stimuli responses were in one area or the other – there were no areas that responded optimally to the morphed features. However, they also found some response for the morphs could be found in either area. But there was not a smooth interpolation between the two areas. Looking deeper, smooth V1 organization is not completely smooth at a lower scale: "the projection of the world into V1 is smooth and continuous on the macroscopic level, but jittery and occasionally discontinuous on the microscopic scale" (Koch, 2004 [52] – pg. 78). Maldonado *et al.* [68] measured the features detected in the pinwheel centers of V1 and found a larger variance of feature types selected for, but not significantly larger bandwidths. This implies that selection of features with low correlated firing can coexist nearby, so averaging of these unrelated nearby features did not necessarily occur.

It is generally assumed that lateral excitation is the impetus for topography to emerge. Supporting results have been found using the computational self-organizing maps [55] (SOM). In cerebral cortex, many lateral connections are clustered close by the neuron from which they originate, to other nearby (i.e., neighboring) neurons. There are also strong long-range connections to neurons that detect similar features (e.g., similar or identical orientations). In the self-organizing maps, an approximation of lateral excitation is isotropic biasing. That is, a firing neuron will excite all the neurons around it equally, as a function of radial distance. But in actual cortex, the close connections are not isotropic, but instead "patchy". Thus an isotropic function of updating emanating from the winner neuron(s) is not biologically accurate. In [73], it was shown that an orientation map with such patchy connectivity can develop by incorporating adaptive, limited-range, lateral connections into an SOM, and using oriented gaussians or natural image patches as stimuli. Lateral excitatory connectivity was also shown to be the cause of "smoothness" of the map, meaning the features represented in a small area (containing a few neurons) tend to be similar. Yet in their mature simulation cortex, excitatory connections between neurons that represent dissimilar stimuli (not very statistically correlated) were not typically present, especially for long-range connections but even for nearby neurons.

Results in Chapter 4 suggest adaptive lateral connections play a crucial role in developing modularity, and coupled with top-down connections can cause modular and abstract representation areas. Since top-down originate from the more abstract association cortices, and the motor areas, the top-down connections carry information that can be used to bias what features are currently important (internal attention) to the task at hand. This idea inspired the approach described in Chapter 5.

Chapter 3

Developing Neuronal Layers

This chapter presents the Lobe Component Analysis (LCA) technique for general-purpose neuronal computation. LCA is a candidate technique for developing local features at any hierarchical layer. The algorithm allows nearly in-place development of each neuronal layer, anywhere in a network.

3.1 Overview

The biologically inspired Lobe Component Analysis incrementally develops a set of optimal pattern detectors from a high-dimensional sensory stream. Each feature detector, which is a vector, is called a "lobe component". The lobe components are optimal in the sense that each is the best representation of the observations that influenced its development. In other words, each feature is selected so that it maximizes the likelihood of its input history. The optimality is implemented via a biologically-inspired Hebbian learning rule. A major advantage of the optimality is that there is no need to manually select (tune) the learning rate for the neurons — each neuron tunes its own learning rate in the best way.

Over learning, each neuron seeks to latch onto a component (also called source, cause, feature, etc.) that "caused" the data (such as a vertical edge), and can be used to "explain" the data. It does this incrementally over a series of observations. To keep other neurons from following the same path, it can laterally inhibit the others; so neurons must find different components. This nonlinear search is accomplished in a nearly-optimal way (allowing for some inappropriate observations to fall into a neuron's history in the initial organization phase between all the components).

The neurons are ensured to learn different features through competition (lateral inhibition). LCA uses k-winners-take-all to approximate the neural competition mechanisms of lateral inhibition. Via the winner-take-all approach, only the most similar lobe component direction to the input will shift to become nearer to the input vector direction. The other neurons do not change, unlike e.g., gradient methods which change all neurons for each sample, therefore LCA has long-term memory.

The technique is purely incremental, and additionally requires no extra storage besides the lobe components themselves. The learning is mostly local, except the approximation of lateral inhibition via k-winners take all. The development is almost in-place.

A neuron's history and the data that fall into its current Voronoi partition may not coincide. We used a mechanism called CCI plasticity, which allows each neuron to "forget" its earlier observations, which may not be in its Voronoi partition anymore. This also provides the network with the ability to deal with nonstationary distributions - the lobe components can eventually adapt to any changes in the environment (such as lighting changes).

3.2 Concepts and Theory

Consider a simple computational model of a neuron (indexed i) having n synaptic inputs. Its firing rate is modeled by

$$y_i = g(v_{i,1}x_1 + v_{i,2}x_2 + ... + v_{i,n}x_n) = g(\mathbf{v}_i \cdot \mathbf{x}), \tag{3.1}$$

where $\mathbf{x} = (x_1, x_2, ..., x_n)$ is the vector of firing rates of each of the n input lines, and the synaptic strength — weight — associated with each input line x_j is $v_{i,j}, j = 1, 2, ..., n$. Traditionally, g has been a sigmoid function. For the analysis and for the experiments provided here, g is not necessary.

A vector of activity on the input lines $\mathbf{x}$ is in the sample space: $\mathbf{x} \in \mathcal{X}$. From the above, note that a neuron's weight vector is also in this space $\mathbf{v}_i \in \mathcal{X}$. A weight vector is called a lobe component.

3.2.1 Belongingness

Belongingness defines the assignment of samples to lobe components. Given a limited resource of c neurons, divide the sample space $\mathcal{X}$ into c mutually nonoverlapping regions, called *lobe regions*:

$$\mathcal{X} = R_1 \cup R_2 \cup ... \cup R_c, \tag{3.2}$$

Each lobe region is represented by a single lobe component. Given any input vector $\mathbf{x}$, it exclusively belongs to a region R_i based on some criteria, e.g. similarity based on some distance metric — $\mathbf{x}$ belongs to $\mathbf{R}_i$ if it is most similar to the representative lobe component $\mathbf{v}_i$ than the other vectors. Belongingness defines a *partition* or *tesselation* of the input space.

3.2.2 Spatial Optimality

Given an assignment of samples to a lobe component i, what is the best way to set $\mathbf{v}_i$?

This, spatial optimality is defined in the following sense. For random input vector $\mathbf{x}$ and matrix of all lobe components $\mathbf{V}$, use notation $\hat{\mathbf{x}}(\mathbf{V})$ to indicate the lobe component for the region an input belongs to. Then, we wish to find the set of lobe components to minimize the expected square approximation error $\mathrm{E}\|\hat{\mathbf{x}}(\mathbf{V}) - \mathbf{x}\|^2$.

$$V^* = (\mathbf{v}_1^*, \mathbf{v}_2^*, ..., \mathbf{v}_c^*) = \arg\min_V \mathrm{E}\|\hat{\mathbf{x}}(\mathbf{V}) - \mathbf{x}\|^2. \tag{3.3}$$

This spatial optimality requires that the spatial resource distribution in the layer is optimal in minimizing the representational error. This distortion is minimized by finding V^* that minimizes the above expression. For a given partition, LCA provides an optimal solution [122]. But if the partition must also be determined, the problem becomes NP-hard [12,81,128]. LCA is an approximate solution by assuming the set of samples falling in a lobe region, based on e.g., minimum inner product difference, is similar to the recent observation history of the representative lobe component. CCI plasticity, defined below, allows old observations to be forgotten.

3.2.3 Temporal Optimality

Incremental (Hebbian) learning algorithms using a single learning rate may find the correct direction of vector change, but will not always take the best "step" towards the goal at each update.

Let $\mathbf{u}(t)$ be the neuronal internal observation (NIO), which for LCA is defined as *response-weighted input*: $\mathbf{u}(t)$

$$\mathbf{u}(t) \overset{\text{def}}{=} \frac{\mathbf{x}(t) \cdot \mathbf{v}(t-1)}{\|\mathbf{v}(t-1)\|}\mathbf{x}(t). \tag{3.4}$$

The synaptic weight vector $\mathbf{v}(t)$ is estimated from a series of observations $U(t) = \{\mathbf{u}(1), \mathbf{u}(2), ..., \mathbf{u}(t)\}$ drawn from a probability density $p(\mathbf{u})$ for this source. Suppose the learning rate η_t is for NIO $\mathbf{u}(t)$ at time t. How can all the learning rates $\eta_1, \eta_2, ..., \eta_t, ...$ be set so that the estimated lobe component $\hat{\mathbf{v}}(t)$ at every time t has the minimum error while the search proceeds along its nonlinear trajectory toward its intended target weight vector $\mathbf{v}^*$?

Let $S(t)$ be the set of all possible estimators for $\mathbf{v}$ from the set of observations $U(t)$. A temporally optimal estimator means that every update at t, $\mathbf{v}$ is spatially optimal over $U(t)$:

$$\text{minimum-error}(t) = \min_{\hat{\mathbf{v}}(U(t)) \in S(t)} \|\hat{\mathbf{v}}(U(t)) - \mathbf{v}^*\|^2, \tag{3.5}$$

for all $t = 1, 2, 3, 4, \ldots$

3.2.4 Solution

According to the theory of Principal Component Analysis (PCA) (e.g., see [49]), the principal component of the conditional covariance matrix $\Sigma_{x,i}$ (conditioned on $\mathbf{x}$ belonging to R_i for lobe component i) is spatially optimal for lobe component i. Now, we need to compute this solution from the data.

First, note $\mathbf{v}_i$ satisfies $\lambda_{i,1}\mathbf{v}_i = \Sigma_{x,i}\mathbf{v}_i$.

Replacing $\Sigma_{x,i}$ by the estimated sample covariance matrix of column vector x, we have

$$\lambda_{i,1}\mathbf{v}_i \approx \frac{1}{n}\sum_{t=1}^{n} \mathbf{x}(t)\mathbf{x}(t)^\top \mathbf{v}_i = \frac{1}{n}\sum_{t=1}^{n} (\mathbf{x}(t) \cdot \mathbf{v}_i)\mathbf{x}(t). \tag{3.6}$$

We can see that the best lobe component vector $\mathbf{v}_i$, scaled by "energy estimate" eigenvalue $\lambda_{i,1}$, can be estimated by the *average* of the input vector $\mathbf{x}(t)$ weighted by the linearized response $\mathbf{x}(t) \cdot \mathbf{v}_i$ whenever $\mathbf{x}(t)$ belongs to R_i.

3.2.5 Incremental Solution

For in-place development, each neuron does not have extra space to store all the training samples $\mathbf{x}(t)$. Instead, it has to update synapses incrementally. The incremental solution to the first principal component follows CCI PCA [125]:

If the i-th neuron $\mathbf{v}_i(t-1)$ at time $t-1$ has already been computed using previous

$t-1$ inputs $\mathbf{x}(1), \mathbf{x}(2), ..., \mathbf{x}(t-1)$, the neuron can be updated into $\mathbf{v}_i(t)$ using the current sample defined from $\mathbf{x}(t)$ as:

$$\mathbf{x}_t = \frac{\mathbf{x}(t) \cdot \mathbf{v}_i(t-1)}{\|\mathbf{v}_i(t-1)\|}\mathbf{x}(t). \tag{3.7}$$

Then Eq. (3.6) states that the lobe component vector is estimated by the average:

$$\lambda_{i,1}\mathbf{v}_i \approx \frac{1}{n}\sum_{t=1}^{n} \mathbf{x}_t. \tag{3.8}$$

Statistical estimation theory reveals that for many distributions (e.g., Gaussian and exponential distributions), the sample mean is the most efficient estimator of the population mean (see, e.g., Theorem 4.1, p. 429-430 of Lehmann [61]). The sample mean is the maximum likelihood estimator for the population mean. In other words, no other estimator in Eq. (3.8) could reach as low of an error given the observations.

Intuitively, each lobe component i develops $\mathbf{v}_i$ to be the expectation of its *response-weighted input*.

3.2.6 CCI Plasticity

The sensory environment of a set of lobe components is not stationary, especially early on. Therefore, the sensory input process is a nonstationary process too. The CCI plasticity technique below which gradually "forgets" old "observations" (which use bad $\mathbf{x}_t$ when t is small) while keeping the estimator quasi-optimally efficient.

The amnesic mean is defined as:

$$\bar{x}^{(t)} = \frac{t-1-\mu(t)}{t}\bar{x}^{(t-1)} + \frac{1+\mu(t)}{t}x_t \tag{3.9}$$

where $\mu(t)$ is the amnesic function depending on t. If $\mu \equiv 0$, the above gives the straight incremental mean. We adopt a three-sectioned profile of $\mu(t)$:

$$\mu(t) = \begin{cases} 0 & \text{if } t \leq t_1, \\ c(t - t_1)/(t_2 - t_1) & \text{if } t_1 < t \leq t_2, \\ c + (t - t_2)/r & \text{if } t_2 < t, \end{cases} \tag{3.10}$$

in which, e.g., $c = 2, r = 2000$. As can be seen above, $\mu(t)$ has three intervals. When t is small, straight incremental average is computed. By the second interval, we hope the lobe component has latched onto a cause, so we increase forgetting rapidly so it can lose the earliest observations before it found its path. Until the third interval, forgetting decreases to a very small (but nonzero) rate, to allow for long-term plasticity.

3.3 Algorithm

The Candid Covariance-free Incremental LCA algorithm incrementally updates c neurons represented by the column vectors $\mathbf{v}_1(t), \mathbf{v}_2(t), ..., \mathbf{v}_c(t)$ from samples $\mathbf{x}(1), \mathbf{x}(2), ...$ The length of $\mathbf{v}_i$ will be the variance of projections of the vectors $\mathbf{x}(t)$ in the i-th region onto $\mathbf{v}_i$.

Initialization — Sequentially initialize c cells using first c inputs: $\mathbf{v}_i(0) = \mathbf{x}(t)$ and set cell-update age $n_i = 1$, for $i = 1, 2, ..., c$.

"Live." For $t = c + 1, c + 2, ...,$ do

1. Pre-competitive response potential. Compute potential for all neurons: For all i with $1 \leq i \leq c$, compute the response[1]:

$$y_i = \frac{\mathbf{x}(t) \cdot \mathbf{v}_i(t-1)}{\|\mathbf{v}_i(t-1)\|}, \tag{3.11}$$

[1] Here we present linear response, but the entire system is nonlinear system due to the top-k mechanism used.

2. Lateral inhibition. Rank $k+1$ top winners so that after ranking, $y_1 \geq y_2 ... \geq y_c$, as ranked responses[2]

Use a linear function to scale the response:

$$y_i' = (y_i - y_{k+1})/(y_1 - y_{k+1}), \tag{3.12}$$

for $i = 1, 2, ..., k$. All other neurons do not fire: $y_i = 0$ for $i = k+1, k+2, ..., c$.

3. Optimal Hebbian learning. Update only the top k winner neurons $\mathbf{v}_j$, for all j in the set of top k winning neurons, using its temporally scheduled plasticity:

$$\mathbf{v}_j(t) = w_1 \mathbf{v}_j(t-1) + w_2 y_j \mathbf{x}(t), \tag{3.13}$$

where the cell's scheduled plasticity is determined automatically by its two update-age dependent weights, called retention rate and learning rate, respectively:

$$w_1 = \frac{n(j) - 1 - \mu(n_j)}{n_j}, w_2 = \frac{1 + \mu(n_j)}{n_j}, \tag{3.14}$$

with $w_1 + w_2 \equiv 1$.

4. Long-term memory. Update the real-valued neuron "age" $n(j)$ only for the winners: $n_j \leftarrow n_j + y_j'$, $j = 1, 2, ..., k$ ($y_j' = 1$ for the top winner). All other neurons i that do not update, keep their age and weights unchanged: $\mathbf{v}_i(t) = \mathbf{v}_i(t-1)$.

[2] For computational efficiency, this non-iterative ranking mechanism replaces repeated iterations that take place among a large number of two-way connected neurons in the same layer.

Figure 3.1: Example of a natural image used in the experiment.

3.4 Experiments

3.4.1 Natural Images

The thirteen images available at `http://www.cis.hut.fi/projects/ica/imageica/` were used as examples of real-world "natural" images, i.e., images whose statistics are representative of the signals we interpret through vision. For the input of each experiment, we incrementally and select a 16 x 16 pixel patch from a random location in a random image, and concatenate it into a column vector. One of the images used in the experiment is shown in Figure 3.1.

Earlier works [?, 65] have already shown that LCA extracts orientation selective features from natural image input.

Figure 3.2 shows the result when using LCA on 256 neurons and 1,500,000 whitened input samples. The lobe components in the image are ordered by the number of times each was the winner during the procedure. The component with the most wins is at the top left of the image grid, and it progresses through each row until the one with the least wins, at the bottom right.

Whitening. Whitening is a preprocessing procedure that decorrelates the inputs. It projects each sample along the principle components, but also adjusting by the scale of each. The whitened sample vector $\hat{\mathbf{x}}$ is computed from the original sample $\mathbf{x}$ as $\hat{\mathbf{x}} = \mathbf{W}\mathbf{x}$, where $\mathbf{W} = \mathbf{V}\mathbf{D}$ is the whitening matrix. $\mathbf{V}$ is the matrix where each principal component $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n$ is a column vector, and $\mathbf{D}$ is a diagonal matrix where the matrix element at row and column i is $\frac{1}{\sqrt{\lambda_i}}$, where λ_i is the eigenvalue of v_i. For a lobe component, it must first be dewhitened in order to display properly. For example, to restore the original input vector, $\mathbf{x} = \mathbf{V}\mathbf{D}^{-1}\hat{\mathbf{x}}$, is the dewhitening procedure.

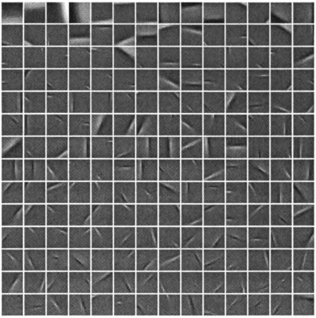


Figure 3.2: Lobe components from natural images (with whitening).

Figure 3.3 shows the result when whitening is not used. The filters are ordered in the same way as the first experiment. In this case, although they do show a preference for certain orientations, most filters are not localized, meaning the receptive fields of each are the entire 16 x 16 window.

Due to the speed of both the algorithm and the programmed implementation, these experiments over 1,500,000 samples took less than 30 minutes on a Pentium M 2.0GHz PC with 1.0GB memory. The overall process is summarized in Figure ??.

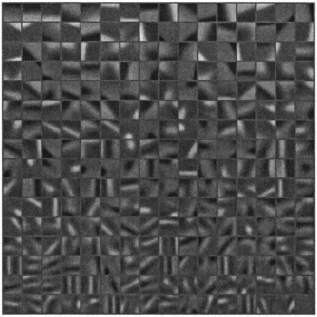


Figure 3.3: Lobe components from natural images (without whitening)

3.4.2 Hebbian Updating

The purpose of this section is to show the effect of LCA's optimality, especially compared to typical Hebbian updating rules, which set a single learning rate manually. These experiments originally done for the LCA article by Weng & Luciw, 2009 [122].

The basic Hebbian form [23,38] for updating the weight vector $\mathbf{v}$ of a neuron is:

$$\Delta \mathbf{v} = \eta y(\mathbf{v}, \mathbf{x})\mathbf{x} \tag{3.15}$$

where $\mathbf{v}$ is the amount of update for the weight vector $\mathbf{v}$ by executing $\mathbf{v} \leftarrow \mathbf{v} + \Delta \mathbf{v}$, η the learning rate, $\mathbf{x}$ the vector input (pre-synaptic activity).

Oja's classic neuron updating algorithm [77] is an algorithm that follows Eq. (3.15) for incrementally computing the first principle component, which is spatially optimal as discussed in earlier sections. Like LCA, its NIO is response-weighted input.

$$\Delta \mathbf{v} = \eta \mathbf{y}(t)(\mathbf{x}(t) - \mathbf{y}(t)\mathbf{v}(t)) \tag{3.16}$$

where $\mathbf{y}(t) = \mathbf{x}^T(t)\mathbf{v}(t)$ is the neuronal response. This version must be used with small η (e.g., $\eta = 10^{-3}$) for stability. If stable, the lengths of the vectors will tend to unit.

A stable two-step version of Eq. (3.16) uses normalization. It aligns directly with Eq. (3.15) and uses time-varying η is:

$$\Delta\mathbf{v} = \eta(t)(\mathbf{x}^T(t)\mathbf{v}(t))\mathbf{x}(t),\ \mathbf{v} \leftarrow \mathbf{v}/\|\mathbf{v}\| \tag{3.17}$$

We called it "Hebbian with TVLR (time-varying learning rate)".

The "dot-product" version of the SOM updating rule [55] (page 115) is also considered as incremental neuronal learning:

$$\mathbf{v}_i(t+1) = \frac{\mathbf{v}_i(t) + \eta(t)\mathbf{x}(t)}{\|\mathbf{v}_i(t) + \eta(t)\mathbf{x}(t)\|} \tag{3.18}$$

where $\mathbf{v}_i$ is the winning component vector at time t. The NIO used by SOM's rule is $\mathbf{u} = \mathbf{x}$ (not weighted by response). Without response-weighting, this updating rule did not perform successfully in these tests.

All of the above use a single learning rate parameter to adapt the neuron weights to each new updating input, and a method to bound the strengths of synaptic efficacies (e.g., vector normalization). CCI LCA weights using the time-varying retention rate $w_1(t)$ and learning rate $w_2(t)$, where $w_1(t) \equiv w_2(t)$, in order to maintain the energy estimate. With the energy gone in the schemes above, there is no way to adjust the learning rate $\eta(t)$ to be equivalent to the CCI scheduling. Therefore, the result of Eqs. (3.16), (3.17) and (4.16) cannot be optimal.

For tuning the time-varying learning rate $\eta(t)$, we used three example suggested [114] learning rates for $\eta(t)$, which were `'linear'`: $\eta(t) = \eta(0)(1 - t/T)$, `'power'`: $\eta(t) = \eta(0)\ (0.005/\eta(0))^{t/T}$ and `'inv'`: $\eta(t) = \eta(0)/(1 + 100t/T)$. The initial learning rate $\eta(0)$ was 0.1 or 0.5. Plasticity parameters for LCA's μ were $t_1 =$

$10, t_2 = 100, c = 5, r = 5000$.

Experiment: Stationary Distributions

The statistics of natural images are known to be highly non-Gaussian [91], and the responses of V1 neurons to natural input have a response profile characterized by high kurtosis. The Laplacian distribution is non-Gaussian and has high kurtosis, so we test estimation of the principle component of Laplacian distributions.

Data was randomly drawn from a d-dimensional Laplacian, with pdf: $f(x|\mu', b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu'|}{b}\right)$. It is a combination of d one-dimensional Laplacians, each of which is best explained with a single vector in the axis direction, with length equal to the variance. All dimensions had zero mean ($\mu' = 0$) and unit variance ($b = 1$). The true sources to be extracted from this distribution are the d directions, with length b.

The number of neurons is set to d, and initialized to random samples drawn from the distribution. For a fair comparison, all methods started from the same initialization. The training length (maximum number of data points) was $T = 10000\ d$, so that each neuron would on average have 10000 updates. Dimension d was 25 or 100, and results were averaged over 50 trials. The results measure how well the neurons' directions match the direction of the true components.

Results are shown in Figs. 3.4 and 3.5. The "SOM" curve shows the best-performing variant among the six different learning rate functions and initial learning rates, as suggested [114]. None of them led to extraction of the true components (the best one uses $\eta(0) = 0.1$ and the `linear` tuning function — in both cases). For Oja's rule with time-varying learning rate, we show only $\eta(0) = 0.1$ since the alternate curves ($\eta(0) = 0.5$) were uniformly worse. These results show the effect of LCA's dual optimality. In 25-dimensions, when LCA has achieved 20% error, the best other Hebbian method has only achieved 60% error. Similarly, in 100-dimensions, when

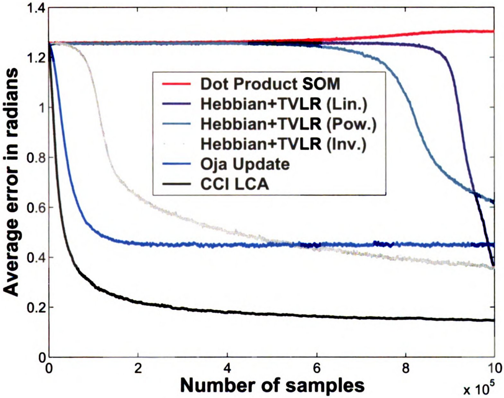


Figure 3.4: Comparison of incremental neuronal updating methods. We compare in 25 and 100 dimensions. This figure shows 100-d results. Methods used were (i) "dot-product" SOM, (ii) Oja's rule with fixed learning rate 10^{-3}, (iii) Standard Hebbian updating with three functions for tuning the time-varying learning rates (TVLR): linear, power, and inverse, and (iv) CCI LCA. LCA, with its temporal optimality, outperforms all other methods. Consider this as a "race" from start (same initialization) to finish (0% error). Note how quickly it achieves short distance to the goal compared with other methods. CCI LCA beats the compared methods. E.g., after 28,500 samples, when LCA has covered 56% distance, the next closest method has only covered 24% distance.

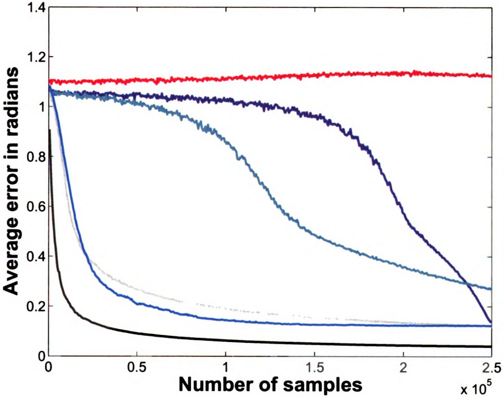


Figure 3.5: Comparison of incremental neuronal updating methods. This shows the 25-d results. At this lower dimension, after 5000 samples, LCA has covered 66% of the distance, while the next closest method has only covered 17% distance.

LCA has achieved 30% error, the best compared method is still at 70% error. The results for LCA will not be perfect due to the nonstationarity that occurs due to self-organization.

Time-varying distributions: plasticity

It is important for an agent to have the capability to adapt to new environments without catastrophic forgetting of what was already learned.

We performed a comparison of how well the best-performing of the algorithms we tested before adapt to a time-varying distribution. We set up a *changing environment* as follows.

This is motivated by how a teacher will emphasize new material to the class, and only more briefly review old material. There are five phases. In the first phase, until time 200,000, the data is drawn from 70 orthogonal Laplacian components that span a 70-dimensional space. In the second phase, from time 200,000 to 400,000, the data is drawn from one of 10 *new* components in the $70 - d$ space (rotated to not lie on axes directions) with a 50% chance *or* from one of the original 70 (using the original rotations) with 50% chance. In the third phase, from time 400,000 to 600,000, the data is drawn from either 10 previously unseen components or the original 70 (50% chance of either). The forth phase, until time 800,000, is similar — 10 more previously unseen components are introduced. In the fifth phase, until T =1,000,000, we draw from all 100 possible components (and each has a 1% probability). We use 100 neurons over all phases (never increases or decreases). There are finally 100 neurons for 100 components, but in early phases we have extra resource (e.g., in phase one, we have 100 neurons for 70 components).

Results, averaged over 50 runs with different rotation matrices for each run, are shown in Fig. 3.6 and Fig. 3.7. LCA outperforms the other two variants — it is better at adaptation, and suffers a more graceful forgetting of data that is not

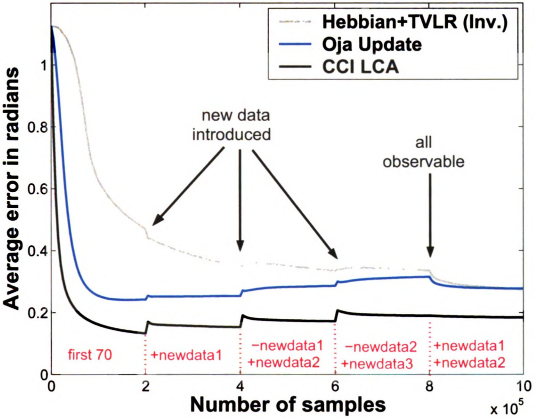


Figure 3.6: Comparison of LCA with two other Hebbian learning variants for a time-varying distribution. This shows average error for all available components. There are 70 available until time 200,000, 80 until 400,000, 90 until 600,000 and 100 until 1,000,000. We expect a slight degradation in overall performance when new data is introduced due to the limited resource always available (100 neurons). The first jump of LCA at $t = 200,000$ is a loss of 3.7% of the distance it had traveled to that point.

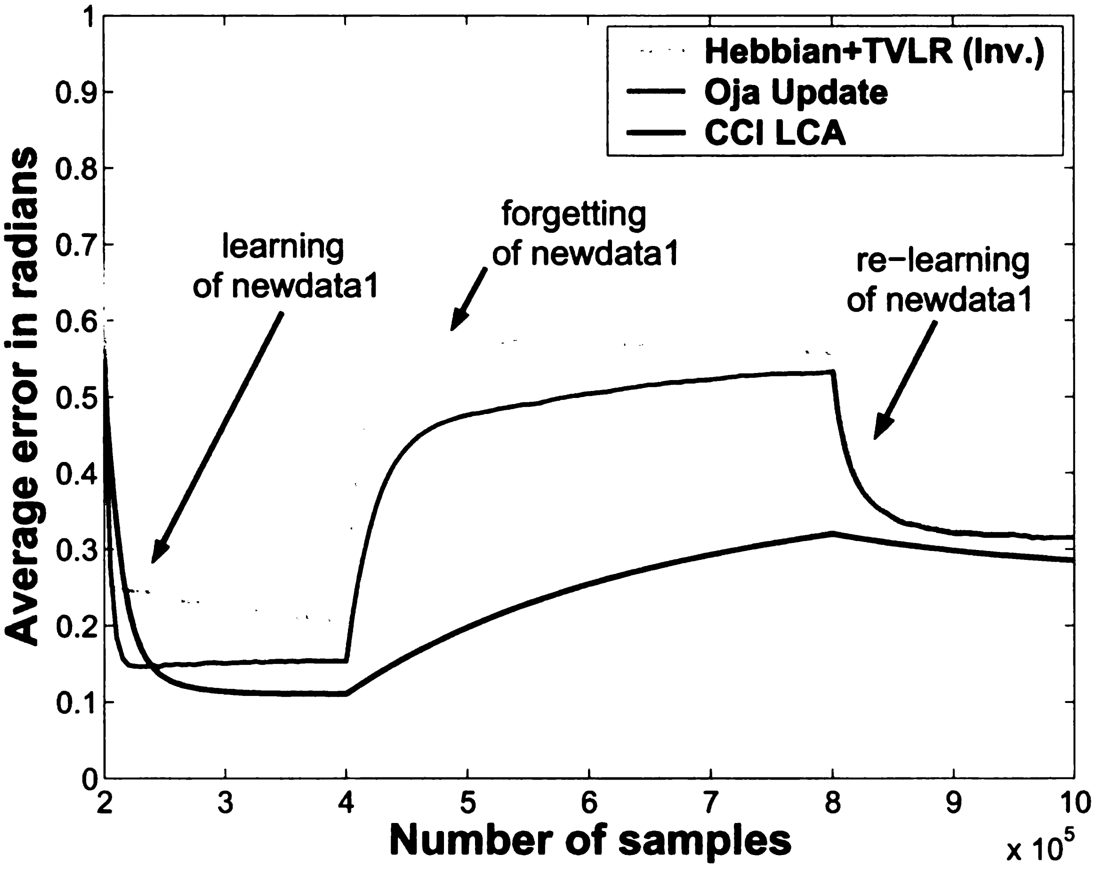


Figure 3.7: Comparison of LCA with two other Hebbian learning variants for a time-varying distribution. This shows how well the neurons adapt to the 10 components added at time 200,000 (called newdata1), and then how well they remember them (they are observed in only the second and fifth phases). Initially, this new data is learned well. At time 400,000, newdata2 begins to be observed, and newdata1 will *not* be observed until time 800,000. Note the "forgetting" of the non-LCA methods in comparison to the more graceful degradation of LCA. The plots focusing on newdata2 and newdata3 are similar.

currently observed. We note that the "re-learning" in the last phase does not match the previously observed performance. This is due to two reasons: the lessening of plasticity for larger neuron ages, and the increasing of the manifold of the data while retaining only a fixed representation resource.

3.5 Bibliographical Notes

The in-place learning concept and the LCA algorithm were introduced in Weng & Zhang, 2006 [124], and used as each layer in our Multi-layer In-place Learning Networks [64, 120]. The MILN-based model of six-layer cerebral cortex [121], which was informed by the work of Felleman & Van Essen [32], Callaway and coworkers [18] and Grossberg [85], used LCA on both its supervised (L2/3) and unsupervised (L4) functional layers. The journal version of LCA [122] was more comprehensive and presented the comparisons with Hebbian learning rules included here. The multilayer models use LCA on each layer.

LCA was inspired by principal components analysis (PCA) and independent component analysis (ICA). Principal components are linearly the most expressive features of a dataset, in terms of least mean square error of the projections [7]. Due to orthogonality constraints, most principal components do not match the causes of the image patch. In ICA, the constraints are not that of orthogonality but independence. With respect to image patches, each patch can be thought of as a combination of independent sources (such as edges at particular orientations).

An ICA approach was applied to *natural images* instead of task-specific views (e.g., from a single object) and developed orientation selective features similar to those found in V1 cortex and Gabor filters [8]. A similar result was found for the non-negative matrix factorization [60] method, which models each image patch as a linear combination of sources, where no component's contribution could be negative.

But as discussed in [66], a problem with the ICA and NNMF methods is that they treat each patch as a linear superposition (weighted sum) of the independent sources, which is not true. So they do not always extract the independent components [66]. It is more appropriate to use *max* instead of *sum*, since real data suggests only one source (i.e., cause) explains each pixel. A formal treatment of the idea is found in Lucke, 2008 [66], who derived an approximate Hebbian learning rule for non-linear component extraction. Earlier, Weng [124] had shown how non-linear extraction could be done with LCA.

Such models are fundamentally trying to set their parameters so that they explain the observed data in the best way. Given a pre-selected model, the theory of maximum likelihood estimation gives an optimality framework about how to set the model's parameters in the best way to explain the data. However, actual maximum likelihood learning is difficult for visual feature extraction, due to the model selection problem, large number of parameters, and local minima, making gradient based methods difficult and initialization-dependent. From maximum likelihood theory, Hinton derived an approximate rule for following the gradient called contrastive divergence, which shows good performance [39]. Contrastive divergence derives V1-like filters when an additional sparse firing criterion is introduced [69], yet the sparseness used has the same problem as ICA since it interprets image patches as linear combinations of independent causes. LCA and Lucke's maximal cause technique [66] both have maximum likelihood interpretations. In LCA, each neuron is meant to "latch on" to a single independent cause early on in learning and via winners-take-all build a history of observations containing that cause. The resulting weight explains that history of observations in the best possible way, which likely reflects the core cause itself (the target). The other network instantiations do not consider this spatiotemporal optimality and thus require learning rates to be small so that neurons do not overshoot their targets. But this dramatically slows down the convergence, as shown by the comparison experiments between LCA and other

Hebbian learning networks here.

Chapter 4

Top-Down Connections for Semantic Self-Organization

The representation learning problem boils down to two criteria: 1. finding the features (neurons) that explain (i.e., represent) experience (data) in the best way, and 2. represent what is more important to the agent more than what is less important, given a limited resource. The first objective was the subject of Chapter 3. This chapter presents a method for dealing with the second criteria.

Networks here utilize three layers, and perform a task of recognition from vision. The third layer is the motor layer, where firing of neurons controls action. The visual sample is labeled by an action module that takes the motor layer's firing, selects the top-firing neuron, and produces the label associated with it. The motor layer itself is not necessarily WTA. These results apply generally to any feature layer with bottom-up and top-down excitatory inputs.

This chapter presents a method for utilizing the top-down connections to develop a biased compression (criteria #2 above). Coupled with lateral excitation, the method develops networks with modular connectivity. Modules contain a motor neuron and multiple feature neurons. The motor neuron and feature neurons project in an excitatory way almost exclusively within the module, while a few "hub" feature neurons have connections to multiple motor neurons[1]. Using hardcoded isotropic lateral excitation, modules additionally become *grouped*, in that all feature neurons are in a single connected location on the feature map. This was called topographic class grouping [64]. Using adaptive lateral excitation [62], modules again emerged, but the grouping was less prevalent and there were no "hub" feature neurons.

4.1 Motivation

It is a core challenge of an autonomous developmental system to automatically generate efficient and effective internal representation from a raw data stream, given a limited resource. This means learning to compress the input so that the important input variation is given higher priority than the unimportant variation. Examples of variation in object recognition include translation, rotation, scale, lighting, etc. For a classification problem, within-class variations (perhaps lighting) are not as important as between-class variations (perhaps shape). What is important to the agent is determined by its behavior, as encapsulated in motor areas of the network. Agents with no experience are to be taught what to do in each case, through supervision and imposed behavior from a teacher.

The cortex does not compress in an unbiased way (e.g., purely reduce redundancy [5]) nor does it derive a compact coding, where, for example, each object is represented by its own neuron. This type of specialization is too expensive (Ito and Gilbert, 1999 [45], p 21). Instead, the coding and compression of information seems to be *selective*. Some information may be lost, but the *important information for completing the task* is likely to be retained. As stated by Barlow, "The best way to code information depends on the use that is to be made of it" (Barlow, 2001 [5]). Behavior and actions must bias organization and coding of earlier sensory

[1] Motor neurons can be considered as within-module hubs [15].

cortical areas. Feedback connections from later levels to earlier levels seem likely causes of such bias. There are just as many, if not more, feedback connections than feedforward connections (Ito and Gilbert, 1999 [45], p 18).

4.2 Concepts and Theory

4.2.1 Three Layer Network

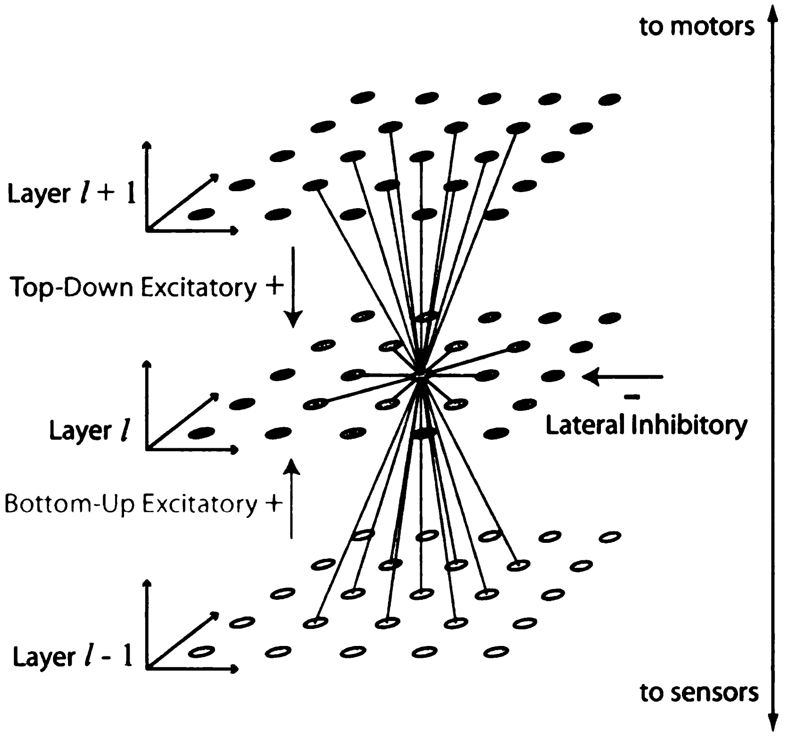


Figure 4.1: In this model, neurons are placed on different layers in a hierarchy - terminating at sensors at the bottom and terminating at motors at the top. Each individual neuron has three types of input projections: bottom-up, lateral, and top-down.

The three-layer network architecture used for theory and experiments in this chapter is shown in Fig. 5.2. There are d pixels ("sensory neurons"), n feature neurons and m motor neurons. Let the number of classes in the input be equal to the number of motor neurons.

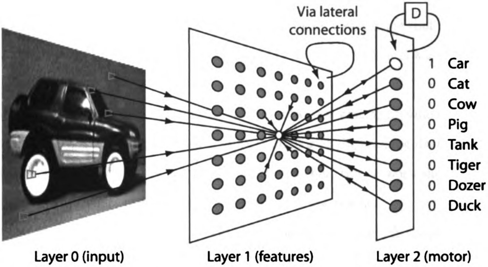


Figure 4.2: The three-layer network structure. The internal layer 1 takes three types of input: bottom-up input from layer 0, top-down input from layer 2 and lateral input from the neurons in the same layer. The top-down input is considered delayed as it is the layer 2 firing from the last time step. The "D" module represents this delay. A circle in a layer represents a neuron. For simplicity, this is a fully connected network: All the neurons in a layer takes input from every neuron in the later layer and the earlier layer. For every neuron (white) in layer 1, its faraway neurons (red) in the same layer are inhibitory (which feed inhibitory signals to the white neuron) and its nearby neurons (green) in the same layer are excitatory (which feed excitatory signals to the white neuron). Neurons connected with inhibitory lateral connections compete so that fewer neurons in layer 1 will win for firing (sparse coding [80]) which leads to sparse neuronal update (only top-k neurons will fire and update). Figure courtesy of Juyang Weng.

Input. Although the networks operate incrementally and in an open-ended fashion, for simplicity consider a set of data. Let S be the supervised (training) set of stimuli, which contains input/output data, where the correct output is provided by a teacher. Any input vector $\mathbf{x}_i$ is for example the raw pixel values of a digital image and its corresponding output $\mathbf{z}_i$ is the motor vector selecting the correct label. For classification, any $\mathbf{x}_i$ is a member of one of m classes. The corresponding output vector from the stimuli set $\mathbf{z}_i$ a vector of zeros except for a single one that denotes its class membership: $(z_i(c_j) = 1) \rightarrow (\mathbf{z}_i \in \text{class } c_j)$. Each dimension of $\mathbf{z}$ is considered a distinct action.

Learning. A network uses data in S to learn so that it is able to provide the correct action for a given test sample: $\mathbf{x} \notin S$, which is similar to the training data. In other words, given a case that was not taught by the teacher, the network should act in the correct way, where correctness is understood by the teacher.

Action. We utilize a hardcoded action production module, which simply uses the index of the motor neuron with the largest firing rate: $\arg\max_{1 \leq i \leq m}\{z_i\}$. It maps the label to a word, that was used in training to teach the class.

Connection types. There are four connections types for a general network layer: bottom-up excitation, top-down excitation, lateral excitation, and lateral inhibition, as shown in Fig. 4.1. In general, neurons on any layer l are connected from the lower and higher layers through excitatory input connections (dendrites and synapses) and excitatory output connections (axons). These grow and are adjusted through learning. Neurons are connected to neurons on the same layer through inhibitory connections. In the model here, the inhibitory connections are not weighted nor learned. They are handled through approximate methods, such as winner-take-all, which are computationally more efficient and easier to deal with. In this section, the lateral excitation is hardcoded and isotropic.

Output and Input Spaces. The firing rate vector is in the output space of

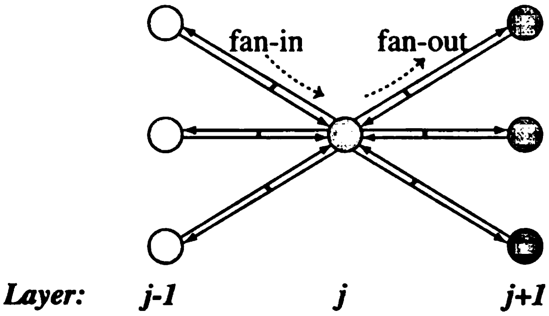


Figure 4.3: How bottom-up and top-down connections coincide in a fully connected network. Looking at one neuron, the fan-in weight vector deals with bottom-up sensitivity while the fan-out weight deals with top-down sensitivity. In the model presented here, the weights are shared among each two-way weight pairs. So, a neuron on layer $j+1$ will have a bottom-up weight the same as the top-down weight of a neuron on layer j.

our layer l, denoted $\mathcal{Y}$. Let the layer $l-1$ closer to the sensors has an output space $\mathcal{X}$ and the layer $l+1$ closer to the motors has an output space $\mathcal{Z}$. Given the diagram Fig. 5.2, we can see that in our case the output spaces of the lowest and highest layers are the same as the sensory input and motor output spaces, respectively. If there's no top-down connections, the input space of layer l is $\mathcal{X}$. On the other hand, if there are top-down connections to layer l, the input space contains all *paired vectors* from $\mathcal{X} \times \mathcal{Z}$, containing both bottom-up and top-down input: $\mathcal{X} \times \mathcal{Z} = \{(\mathbf{x}, \mathbf{z}) | \mathbf{x} \in \mathcal{X}, \mathbf{z} \in \mathcal{Z}\}$.

Weights. The bottom-up weights of the feature neurons are column vectors in weight matrix $\mathbf{V}$, which has n columns. A key aspect of this model is the bottom-up weights to layer $l+1$ and the top-down weights to layer l are *shared* (see Fig. 5.3). Let the bottom-up weight matrix to layer $l+1$ be $\mathbf{W}$, and the top-down weight matrix to layer l is $\mathbf{M} = \mathbf{W}^T$.

4.2.2 Relevant Input is Correlated with Abstract Context

An extraordinary amount of the information we experience can be considered *irrelevant.* Knowledge of what is relevant and what is irrelevant is especially important during learning. This model proposes that relevance in the input is findable by observing what is correlated with *imposed* action. For example, if a child sees the shape of the letter "A" on a flashcard, on television, on a sign, etc. and a teacher gets the child to speak "A" in each case, then the relevant information (the shape of the letter) does not change too much over the different inputs, but other information (the surrounding visual scene) changes a lot.

For the network described here, let the image input space $\mathcal{X}$ be made up of relevant subspace $\mathcal{R}$ and irrelevant subspace $\mathcal{I}$, where relevance is in terms of the data class labels: $\mathcal{X} = \mathcal{I} \times \mathcal{R}$ (See Fig. 4.4). Projecting samples into the irrelevant subspace will not aid in class discrimination, but it will help to do so with the relevant subspace.

Including the higher-layer (more abstract) context space, the relevant subspace is defined as $\mathcal{R} \times \mathcal{Z}$:

$$\mathcal{X} \times \mathcal{Z} = (\mathcal{I} \times \mathcal{R}) \times \mathcal{Z} = \mathcal{I} \times (\mathcal{R} \times \mathcal{Z})$$

Let an abstract-boosted input vector $\mathbf{p}$ to layer l be a combination of a output of layers $l-1$ and $l+1$: $\mathbf{p} = (\mathbf{x}, \mathbf{z})$. We wish to use the higher-layer part to uncover the relevant subspace $\mathcal{R}$. Self-organization of neurons in the space $\mathcal{R} \times \mathcal{Z}$ will lead to discriminant features (neurons), through the use of this abstract-boosted input.

4.2.3 Null Space

Linear Discriminant Analysis (LDA) defines relevant information as the set of projected data onto a linear subspace that can be used to most accurately classify the data, for any linear subspace of the same dimension.

For the random input vector $\mathbf{p}$, the *within-class scatter* defines the average covariance of a classes samples:

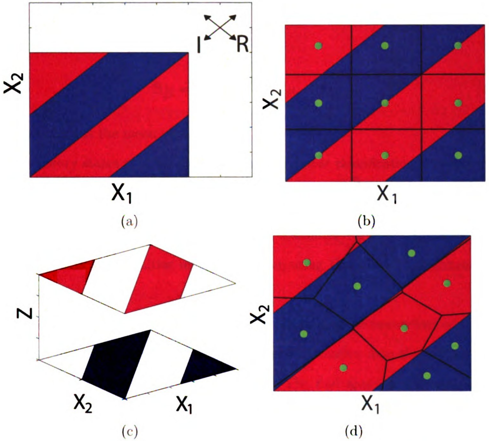


Figure 4.4: Self-organization with motor-boosted distances leads to partitions that separate the classes better. (a) There are two bottom-up dimensions x_1 and x_2. Samples falling in the blue area are from one class and those falling in the red area are another class (assume uniform densities). The "relevant" and "irrelevant" dimension are shown by the upper right axes, which are here linear. (b) The effect of self-organization using nine neurons in the bottom-up space only. Observe from the resulting partitions that the firing class entropy of the neurons will be high, meaning they are more class-mixed. (c) Boosting the data with motor information, which here is shown as a single extra dimension instead of two (for visualization) (d) The effect of self-organization in the boosted space, and embedding back into two dimensions. Note how the partition boundaries now line up with the class boundaries and how the data that falls into a given partition is mostly from the same class (low entropy).

$$\mathbf{S}_W = \sum_{i=1}^{m} \sum_{\mathbf{p} \in c_i} (\mathbf{p} - \mu_i)^T (\mathbf{p} - \mu_i) \tag{4.1}$$

where $\mu_i = E[\mathbf{p} \in c_i]$ indicates the within-class mean for class i. The *between-class scatter* defines the covariance of the class means:

$$\mathbf{S}_B = \sum_{i=1}^{m} (\mu_i - \mu)^T (\mu_i - \mu) \tag{4.2}$$

where $\mu = E[\mathbf{p}]$ is the mean of $\mathbf{p}$.

LDA theory states that the best subspace for linear classification is spanned by the eigenvectors associated with the largest eigenvalues of $\frac{\mathbf{S}_B}{\mathbf{S}_W}$. The most discriminant feature is the first eigenvector, which will maximize $\frac{\mathbf{v}^T \mathbf{S}_B \mathbf{v}}{\mathbf{v}^T \mathbf{S}_W \mathbf{v}}$. However, there are many practical difficulties involved in computing this given high-dimensional data [97].

Instead of dealing with the above expression directly, some researchers advocated to attempt to project the between-class scatter information into the null space of $\mathbf{S}_W$ and deriving features in this space [19]. The null space of the within-class scatter is very powerful. Intuitively, if a direction $\mathbf{v}$ exists where $\mathbf{S}_W\mathbf{v} = 0$ and $\mathbf{S}_B\mathbf{v} \neq 0$, perfect classification can be attained using this $\mathbf{v}$. The null space may not exist non-trivially with the original data.

As formulated above, the top-down part of the abstract-boosted data is in the null space of the within-class scatter matrix. If we let matrix $\mathbf{Z}$ be an orthogonal basis of layer $l+1$'s output space $\mathcal{Z}$, it can be seen that $\mathbf{S}_W\mathbf{Z} = \mathbf{0}$. Additionally, if layer $l+1$ is a classification motor layer, where each dimension represents a different class and each sample only has one class, then $\mathbf{S}_B\mathbf{Z} \neq \mathbf{0}$. The space $\mathcal{Z}$ is orthogonal to the subspace defined by the within-class scatter, and can trivially be used for perfect classification of the training data.

There are two powerful properties associated with $\mathcal{X} \times \mathcal{Z}$ for a classification motor, which should be apparent:

Property 4.2.1. *Class Separateness: Since each motor dimension is associated with a different class, distributions for any two classes are guaranteed to be separated in* $\mathcal{X} \times \mathcal{Z}$.

Property 4.2.2. *Similarity Bias: Any two samples from different classes have a greater distance between one another in* $\mathcal{X} \times \mathcal{Z}$*, but any two samples from the same class have the same distance in* $\mathcal{X} \times \mathcal{Z}$ *as in* $\mathcal{X}$.

We will exploit these properties for a biased self-organization[2]. After learning, a non-imposed sample is not in $\mathcal{X} \times \mathcal{Z}$, but is in $\mathcal{X}$. We wish to achieve a good self-organization using $\mathcal{X} \times \mathcal{Z}$ so performance holds up when $\mathcal{Z}$ is no longer available.

4.2.4 Weighted Semantic Similarity

It is useful to be able to control the relative influence of bottom-up and top-down. When a layer uses paired input $\mathbf{p} = (\mathbf{x}, \mathbf{z})$, the influence of the motors may not be sufficient, since the dimension of the input is typically large (e.g., a 40 row and 40 column digital image gives 1600 dimensions) and the dimension of the output is typically not (e.g., 10 classes). Instead, normalize each input source and control the relative influence by

$$\mathbf{p} \leftarrow \left(\alpha \frac{\mathbf{x}}{\|\mathbf{x}\|}, \beta \frac{\mathbf{z}}{\|\mathbf{z}\|}\right) \tag{4.3}$$

where it is expected that $\alpha + \beta = 1$. Setting $\alpha = \beta = 0.5$ gives the bottom-up and top-down spaces equal influence. Raising β will increase the between class scatter by increasing the distance between classes in $\mathcal{X} \times \mathcal{Z}$, as seen in Fig. 4.4. The normalization places each class distribution on its own unit sphere (when dimension

[2] Here, I focus on classification. Solgi has derived results for the general regression case [94].

of layer $l-1$ is more than two), separate from other classes in $\mathcal{X} \times \mathcal{Z}$. Normalization constrains the maximum distance between classes in $\mathcal{X} \times \mathcal{Z}$. It also reduces the effect of high-varying dimensions within each class.

Bottom-up and lateral information-based similarity is purely physical and is non-semantic. Top-down-based similarity is based on more abstract information. If a network is shown images of a bench and a desk chair, and in each time the teacher says "chair, chair", then by the above, α and β can be set so that the inner product angle difference between the two images is very low; even zero (if $\alpha = 0$ and $\beta = 1$).

4.2.5 Smoothness

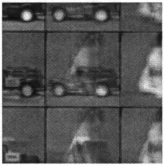


Figure 4.5: A layer-one weight vector, around other neighbor weight vectors, viewed as images, of a neuron exhibiting "harmful" interpolation through 3x3 updating.

Some sort of regularization is necessary. Without it, the placement of neuron feature vectors may not correspond well to the stimuli density. One method is by using the idea of neighborhood and neighborhood updating, from the Self-Organizing Maps [55]. This smoothness approximates lateral excitatory connections which are more dense closer to the neuron of origin. We use 3×3 updating, meaning the winner neuron will update its weights and so will the neurons adjacent to it.

It is beneficial for most of the neighborhood pulling to be within classes. Smoothness is useful for generalization, but this isotropic type could be harmful. Consider a neuron with two neighbors that fire often. This neuron is *pulled* in $\mathcal{X} \times \mathcal{Z}$, by both of its neighbors, without regard to what it actually represents. Pulling places it in between what the neighbor neurons represent. This could be useful if they represent a single class, as it would average into a variation that might generalize well for that class. But if they represent different classes, the averaging effect may lead to a representation of something that wouldn't actually ever be experienced (see Fig. 4.5).

Topographic Class Grouping, discussed later, will emerge via 3×3 updating along with abstract-boosted distance. That leads to helpful interpolation within the class groups, while neurons in different class groups are protected from one another by border neurons. This will be discussed in more detail in Section 4.4.

4.3 Algorithm

Here is presented the algorithm for incremental motor-boosted self-organization for a three layer network. This version handles each sample separately. It is linear time complexity in the number of neurons.

LCA is used on each layer for self-organization, as follows:

$$(\mathbf{y}, \mathbf{V}, \mathbf{0}, \mathbf{a}^{(1)}) \leftarrow f_{LCA}(\mathbf{x}, \mathbf{z} | \mathbf{V}, \mathbf{M}, \mathbf{a}^{(1)}) \quad (4.4)$$

$$(\mathbf{z}, \mathbf{W}, \mathbf{M}, \mathbf{a}^{(2)}) \leftarrow f_{LCA}(\mathbf{y}, \mathbf{0} | \mathbf{W}, \mathbf{0}, \mathbf{a}^{(2)}) \quad (4.5)$$

The feature layer's input is from two sources: bottom-up and top-down stimuli ($\mathbf{x}$ and $\mathbf{z}$). It uses the bottom-up weights $\mathbf{V} = (\mathbf{v_1}, ..., \mathbf{v_n})$, and top-down weights $\mathbf{M} = (\mathbf{m_1}, ..., \mathbf{m_n})$. It updates $\mathbf{V}$ and the neurons' ages $\mathbf{a}^{(1)}$, and outputs the firing rate vector $\mathbf{y}$.

The motor layer takes the firing rate $\mathbf{y}$ as input and uses the bottom-up weights to the motor $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_c)$. It outputs $\mathbf{z}$, which is fed back as the top-down input to Layer 1. In training, $\mathbf{z}$ is imposed.

In this version, the bottom-up weights to the motor layer and the top-down weights to the feature layer are shared: $\mathbf{W} = \mathbf{M}^T$.

For the per-neuron normalization discussed earlier, let $\hat{\mathbf{V}}$, $\hat{\mathbf{W}}$, and $\hat{\mathbf{M}}$ be filled and maintained throughout the below algorithm with normalized columns of $\mathbf{V}$, $\mathbf{W}$ and $\mathbf{M}$ (or zero vectors when appropriate). For example: $\hat{\mathbf{V}} = \left(\frac{\mathbf{v}_1}{\|\mathbf{v}_1\|_2}, ..., \frac{\mathbf{v}_c}{\|\mathbf{v}_n\|}\right) = (\hat{\mathbf{v}}_1, ..., \hat{\mathbf{v}}_n)$.

Initialization[3] – Sequentially initialize n synaptic weight vectors (columns of $\mathbf{V}$) using first n stimuli: $\mathbf{v}_t = \mathbf{x}(t)$ for $t = 1, 2, ..., n$. Place these feature neurons evenly spaced on the d-dimensional feature plane so the distance between non-diagonal neighbors is one. Fill the weights between layer-one and two (matrices $\mathbf{W}$ and $\mathbf{M}$) with zeros. Set the initial cell ages to one.

1. Sense. Draw a stimulus $\mathbf{x}$ from S randomly. Impose the correct action l: the motor vector is set to $z_l = 1$ and $z_i = 0, \forall i \neq l$.

2. Pre-response. Compute pre-competitive potential vector $\hat{\mathbf{y}}$ for layer-one, using both bottom-up and top-down:

$$\hat{\mathbf{y}} \leftarrow \alpha \frac{\mathbf{x}}{\|\mathbf{x}\|} \hat{\mathbf{V}} + \beta \frac{\mathbf{z}}{\|\mathbf{z}\|} \hat{\mathbf{M}}. \tag{4.6}$$

3. Lateral-Inhibition. To compute the firing vector $\mathbf{y}$ for the feature layer using lateral inhibition: Set all neurons' firing to zero except the highest k_1 pre-responses ($k_1 = 1$ for winner-take all). If there are ties, break them randomly. Relatively scale the firing neurons' responses as follows: Rank the elements of $\hat{\mathbf{y}}$. Let s_1 be the highest value, s_k be the k-th highest and s_{k+1} be the $k+1$-th highest. Then set each neuron response as:

[3] This initialization method is a key for fast development; it is better if the first n samples contain some from all different classes.

$$y_i \leftarrow \begin{cases} \dfrac{\hat{y}_i - s_{k+1}}{s_1 - s_{k+1}}, & \text{if } \hat{y}_i \geq s_k \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

4. Optimal Hebbian Learning for Feature Neurons. Each layer-one firing neuron, corresponding to a nonzero element of $\mathbf{y}$, updates its bottom-up weights to be more similar to the bottom-up stimulus $\mathbf{x}$. **For each updating neuron** $i = 1, ..., k_1$, first set its updating weight based on this neuron's age: $\gamma_i \leftarrow \dfrac{1 + \rho(a_i^{(1)})}{a_i^{(1)}}$, where ρ is the CCI plasticity function to control fast adaptation and long-term plasticity [122]. Then, update the bottom-up weights:

$$\mathbf{v}_i \leftarrow (1 - \gamma_i)\mathbf{v}_i + \gamma_i \, y_i \, \mathbf{x} \tag{4.8}$$

and this neuron's age $a_i^{(1)} \leftarrow a_i^{(1)} + 1$.

5. Lateral Excitation. Each winning neuron's neighbors in a 3 × 3 neighborhood update via an approximation of lateral excitation. For each "neighbor" neuron j that hasn't already updated, measure the distance between it and the nearest firing neuron i as $r_{i,j}$. Set its updating weight based on each neuron's age and distance: $\gamma_j \leftarrow \dfrac{1 + \rho(a_j^{(1)})}{a_j^{(1)}} \left(1 - r_{i,j}/2\right)$, and update the bottom-up weights for the neighbors not in the winning set:

$$\mathbf{v}_j \leftarrow (1 - \gamma_j)\mathbf{v}_j + \gamma_j \, y_j \, \mathbf{x}, \tag{4.9}$$

as well as their ages: $a_j^{(1)} \leftarrow a_j^{(1)} + r_{i,j}$.

6. Optimal Hebbian Learning for Motor Neurons. On the motor layer, the winning motor neuron is already known as l, since the action was imposed. The updating follows LCA. Set its updating weight: $\gamma \leftarrow \dfrac{1 + \rho(a_l^{(2)})}{a_l^{(2)}}$, then update this motor neuron's bottom-up weights:

$$\mathbf{w}_l \leftarrow (1-\gamma)\mathbf{w}_l + \gamma\, \mathbf{y}, \tag{4.10}$$

and update its age: $a_l^{(2)} \leftarrow a_l^{(2)} + 1$.

7. Updating Top-Down Weights. Update the top-down weight matrix: $\mathbf{M} \leftarrow \mathbf{W^T}$.

4.4 Topographic Class Grouping

The above algorithm leads to Topographic Class Grouping in many instances. TCG means neurons that represent a particular class are grouped together on the neuronal plane. The importance of TCG is that it allows us to use 3 × 3 updating to regularize the self-organizing network, but we reduce motor interference. With grouping, interference typically occurs within the same class, where it possible becomes useful for generalization along the class' manifold.

To clarify how TCG can emerge, we present a set of conditions sufficient for TCG to emerge in some cases. Whether or not TCG emerges depends very much on the class distributions. It may not work if there's a multi-modal distribution for a single class, or highly overlapping distributions for two classes. This version is based on small groups that grow into larger groups.

First define three types of top-down connectivity: *linked*, *border*, and *unassociated*.

- If $\|\hat{\mathbf{m}}_i\|_\infty > 1-\epsilon$, and $\hat{\mathbf{m}}_{i,j} > 1-\epsilon$ (for some small ϵ), neuron i is linked to class j. A neuron can only be linked to one class. Linked neurons became so since they mostly won for only a single class.

- A neuron with its updating age zero (has never won) is unassociated.

- Otherwise, neuron i is a class-mixed border neuron — border neurons have updated for samples from multiple classes, but not enough to link to any class.

Now define the TCG property:

Property 4.4.1. *Topographic Class Grouping. A network has this property if there is a path from every neuron linked to a class to any other neuron linked to the same class, and there is at least one neuron linked to every class. A path is a sequence of neurons in which consecutive neurons are adjacent.*

Lemma 4.4.2. *If there is at least one linked neuron for the current sample's class, β can be set so that no unassociated neuron can win.*

The pre-competitive response computation for any neuron i is

$$\hat{y}_i \leftarrow \alpha \frac{\mathbf{x}}{\|\mathbf{x}\|} \hat{\mathbf{v}}_i + \beta \frac{\mathbf{z}}{\|\mathbf{z}\|} \hat{\mathbf{m}}_i. \tag{4.11}$$

The top-down part $y_{t,i} = \beta \frac{\mathbf{z}}{\|\mathbf{z}\|} \hat{\mathbf{m}}_i$ will be close to β for a neuron linked to the current training class. For unassociated neurons and neurons linked to other classes, $y_{t,i} = 0$. If $\beta > \alpha + \epsilon$, then a neuron that is linked to the current imposed class will always have higher pre-competitive potentials than neurons that are linked to other classes.

We can establish TCG initially (in a somewhat restrictive manner), by the following:

Base Case: Given the first n samples from n different classes, if the first n winners are unassociated, the network has TCG. After a neuron wins for the first time, it and its neighbors are linked to the current class (see algorithm); they also update bottom-up weights. A simple way to ensure the first n winners are different is to train from the initialization set, using a single sample from each class. Then, all the winners will be different if the first n samples are different in $\mathcal{X}$.

Here is more detail about what happens for the first sample from any class c_i:

1. The first sample from a class c_i is imposed and the appropriate motor neuron i is imposed for the first time.

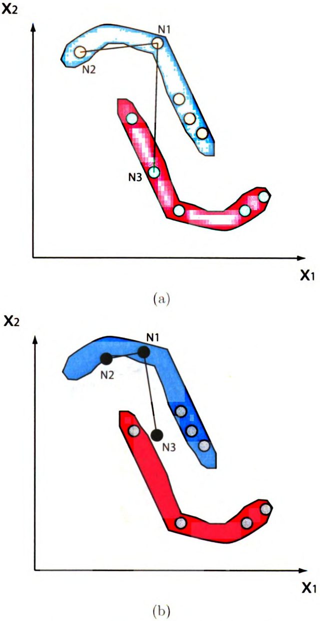
2. The motor neuron c_i updates the appropriate column of $\mathbf{W}$ with a learning rate of one, thus becoming sensitive to the feature layer's firing pattern $\mathbf{y}$ exactly. This firing pattern has k_1 neurons and their neighbor neurons firing (nonzero). Let $k_1 = 1$, so there will only be one neuron 3×3 group.

3. The top-down matrix $\mathbf{M}$ is updated based on $\mathbf{W}$. Then the neurons that fired in the feature layer become the only ones with a nonzero top-down weight from motor neuron i. This establishes the initial group for c_i.

Hypothesis: Assume we have a network with TCG. **Step**: Now, a new sample is input from an arbitrary class. TCG will not be violated after the update if the winner is a linked neuron to the current class, as long as any growing group does not bisect an existing group (see Fig. 4.8). This is so since the linked winner cannot convert any neighbor into a neuron linked to a different class. How can we ensure the winner is always a linked neuron? By Lemma 4.4.2, we can set β so no unassociated neurons will win. As for the border neurons, it's not possible to guarantee this over all data distributions. From LCA theory, we know the border neuron is an average of its response weighted input conditioned on its firing. For sensible data, a border neuron represents a mixture of different classes, and should lie in a very low density area, thus not getting much bottom-up support.

Groups will spread and grow since neighbor updating does not pull the neurons all the way to the winner. The pulled neighbors end up somewhere else in the density. Figure 4.6 and Figure 4.7 illustrates the incremental grouping and growing process with simple class areas in 2D and a 1D neuronal array.

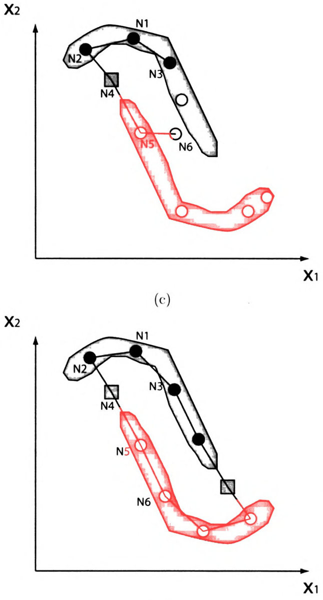
TCG is not guaranteed to emerge, even for the conditions above. A bisection is seen in Fig. 4.8. It should be apparent that the extra degree of freedom allows a growing class group to break an already existing one in two. But, this might only happen if one class is trained at a time.



(a)

(b)

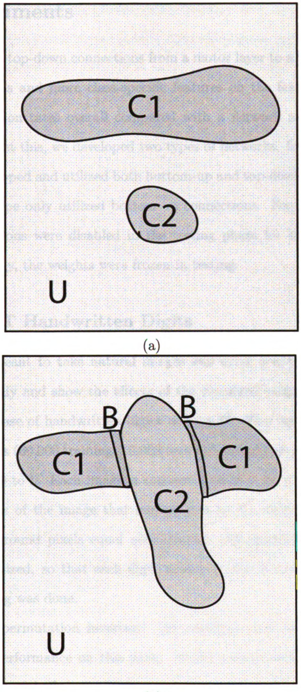
Figure 4.6: Topographic class grouping with a 1D neuron array in 2D input space. The red area contains samples from class one, and the blue area contains samples from class two. The 10 neurons' bottom-up vector are circles or squares. Their top-down membership is shown by the color or shape: Gray neurons are unassociated, black neurons are linked to class two and white neurons linked to class one. Square neurons are border neurons. To better understand how TCG emerges, we provide the following four cases. (a) After initialization, all neurons are unassociated. Only three neurons are drawn to show they are neighbors. Other neighbor connections are not shown yet, for clarity. (b) Neuron N1 has won for a nearby sample, becomes linked to class two. Its neighbors are pulled towards it and also link to class two. Note how N3 is actually pulled into the between-class "chasm", and not onto the class distribution.



(c)

(d)

Figure 4.7: Topographic class grouping with a 1D neuron array in 2D input space. This is a continuation of the last figure. (c) Over wins by N1, N2 and N5, self-organization occurred through neighbor pulling. N4 has become a border neuron, and N6 is starting to be pulled. (d) A final organization, uncovering the relevant dimension. For this data, the relevant dimension is not linear: it is through the center area of each class distribution, lengthwise. The final neuron organization mirrors this, and the neurons have organized along the relevant dimension.



(a)

(b)

Figure 4.8: This shows why TCG cannot be guaranteed in 2D in general. (a) After much sampling of class one, and none of class two, the class one area has grown. (b) Further, after much sampling of class two and none more of class one, the class two area grows, and happens to cut through the class one area (leaving two buffer zones). Here, TCG did not emerge.

4.5 Experiments

It is proposed that top-down connections from a motor layer to a feature layer cause grouped class areas and more class-specific features on the feature layer, leading to higher recognition rates overall compared with a network not using top-down connections. To test this, we developed two types of networks. In the first type, the feature layer developed and utilized both bottom-up and top-down connections. The second network type only utilized bottom-up connections. For a fair comparison, top-down connections were disabled in the testing phase for both network types tested. Additionally, the weights were frozen in testing.

4.5.1 MNIST Handwritten Digits

The network is meant to take natural images and other real-world modalities as input. But to study and show the effects of the discussed properties more clearly, the MNIST database of handwritten digits was used[4]. This well-known dataset of 70,000 total images (60,000 training, 10,000 testing) contains 10 classes of handwritten digits - from 0 to 9. Each image is composed of $28 \times 28 = 784$ pixel intensity values. The pixels of the image that corresponds to the digit are nonzero intensities. The background pixels equal zero (black). All images have already been translation-normalized, so that each digit resides in the center of the image. No other preprocessing was done.

The greedy, "permutation invariant" [39], method used here is not expected to top the best performance on this data. In the current form, the network will evaluate each digit globally - similarity is based on the positions of the non-black pixels. This makes the network very susceptible to translation, rotation, and scale invariance. A localized, windowed, method and/or a method that trains via error

[4] It is available at http://yann.lecun.com/exdb/mnist/.

gradient following is probably necessary. Other methods (e.g. [58], with local analysis and deformations) are better suited for the digit recognition problem.

Results

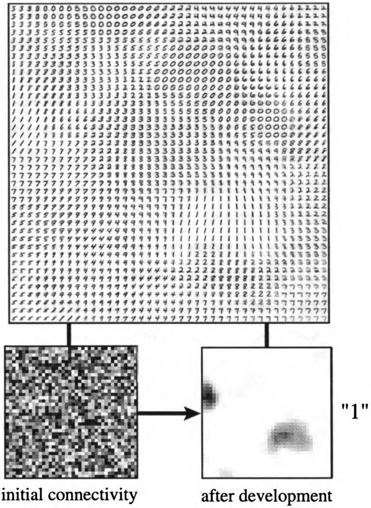


Figure 4.9: MNIST data represented in a network with 40 × 40 neurons, trained without top-down connections.

Fig. 4.9 shows the layer-1 neurons in a 40 × 40 grid trained with all training samples, but not using top-down input. There are areas which can be seen to correspond to a particular class, but there is no guarantee that the area of a single class is connected in the topographic map.

The lower part of Fig. 4.9 shows the development of the bottom-up weights of the motor neuron corresponding to the digit-1 class. The darker the intensity, the larger the weight. Due to the sparseness parameter $k = 1$, the inter-level pathways have become sparse and refined (most connections have been pruned). As shown, invariance is achieved by the positive weights from the corresponding digit "1" neurons to the corresponding output neuron. Therefore, the within-class invariance shown at the output layer can be learned from multiple regions in the previous layer.

The TCG algorithm is quite powerful in its class separation ability. Single groups per class result even for very similar bottom-up inputs, such as the handwritten digits "4" and "9", as seen in Fig. 4.10. As summarized in Table 4.2, when top-down projections are used the error rate is significantly lower (21.3% down to 7.7%) for this limited set of 10 × 10 neurons. In the large scale tests, all ten classes are used and error rate constantly decreased when top-down projections were used, at grid sizes of 10 × 10, 20 × 20 and 40 × 40. Figure 4.11 (a) shows the result for all 10 classes. Each class takes enough grid space within which to explore its own within-class variations with less interference from other classes. Neurons are more "pure", when top-down supervision is used. Purity is measured by entropy (see Appendix). Figure 4.11 part (b) shows the corresponding probability maps of the ten neurons of part (a), and part (c) shows the probability maps for a result when $\beta = 0$. Table 4.2 summarizes results for all tests. Top-down connections led to a better performance for the same size map compared with when no top-down was used. The best overall performance on the data was for a 100×100 map with $\beta = 0.3$, which reached 2.97%. This is on the same level as Hinton's 2.49%, for the contrastive divergence technique [39]. It is a fair comparison since both techniques are greedy and permutation invariant. Using the gradient to fine tune the representation later is possible (as Hinton did).

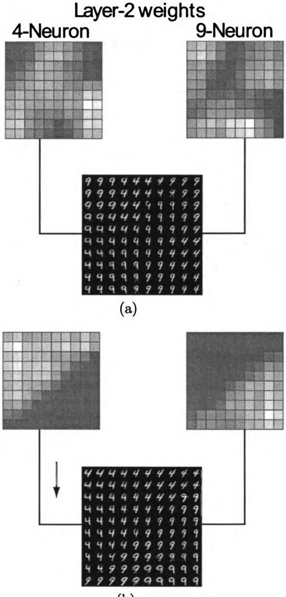
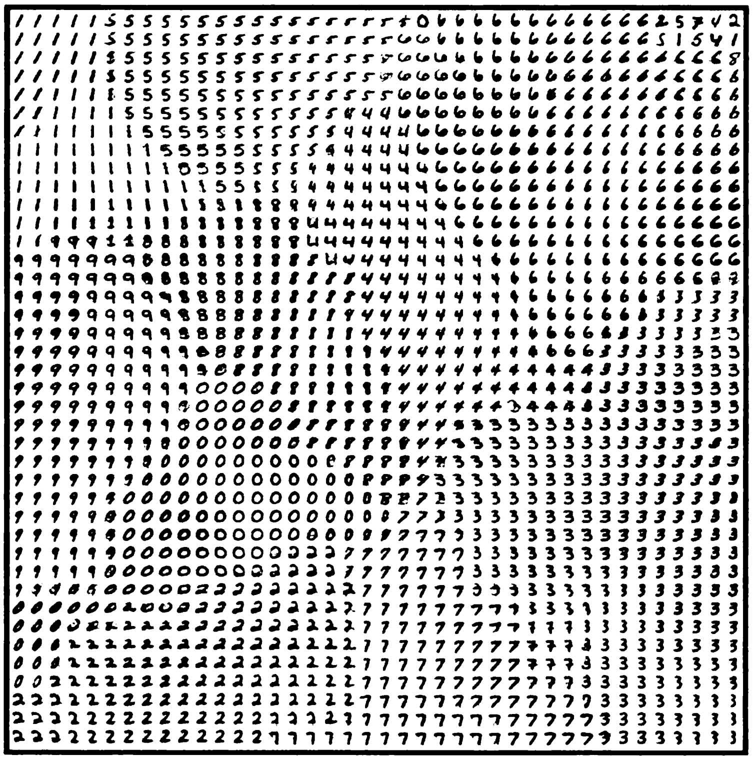


Figure 4.10: The handwritten digits “4” and “9” from the MNIST digit database [58] are very similar from the bottom-up. (a) Result after self-organizing using no motor-boosting. The 100 weight vectors are viewed as images below and the two weight vectors for each motor shown above. White means a stronger weight. The organization is class-mixed. (b) After self-organization using motor-boosted distance (weight of 0.3). Each class is individually grouped in the feature layer, and the averaging of each feature will be within the same class.



(a)

Figure 4.11: (a) Weights after development of a 40 × 40 grid when top-down connections were used. The map has organized so that each class is located within a specific area.

Table 4.1: Summary of results of training networks on MNIST data. Each result is averaged over 5 trials.

Grid size	Top-down (β)	Class set	Train	Test	Error rate	Entropy
10 × 10	no	{4,9}	2000	500	21.3%	0.75
10 × 10	yes	{4,9}	2000	500	**7.7%**	**0.34**
10 × 10	no	0-9	60000	10000	25.6%	0.65
10 × 10	yes	0-9	60000	10000	**16.1%**	**0.36**
20 × 20	no	0-9	60000	10000	13.2%	0.46
20 × 20	yes	0-9	60000	10000	**8.6%**	**0.22**
40 × 40	no	0-9	60000	10000	8.3%	0.34
40 × 40	yes	0-9	60000	10000	**5.7%**	**0.10**

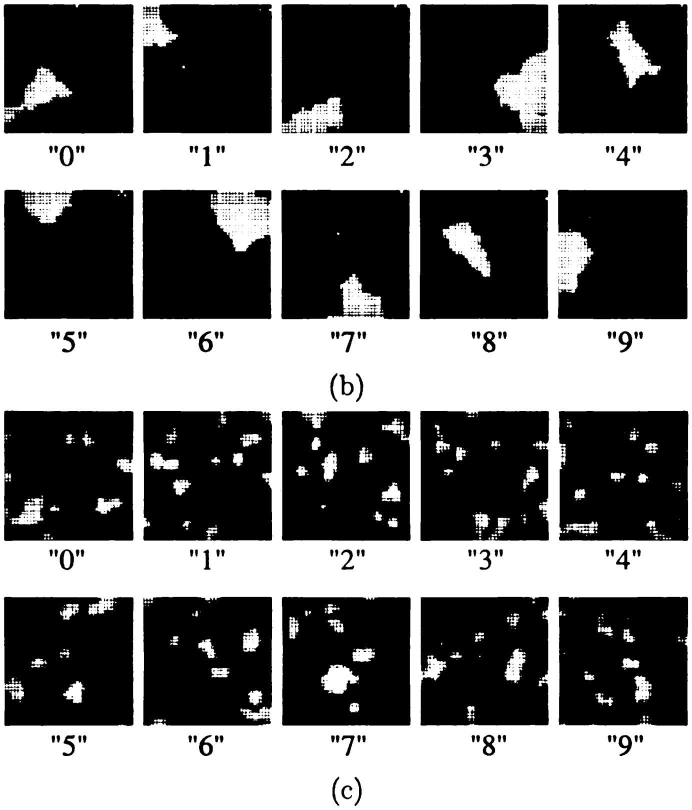


Figure 4.12: This corresponds to the last figure. (b) Probability of each neuron to signify a certain class, from previous updates, when $\beta = 0.3$ is strong (corresponding to (a)). (c) Probability maps for a different test when $\beta = 0$.

4.5.2 MSU-25 Objects

For the dataset, 25 toy objects were selected (see Fig. 4.17). To collect the images, each object was placed on a rotatable surface a few feet in front of a camera. The surface was rotated at a slow rate, and the camera captured images sequentially and automatically, while the operator ensured that the object appeared roughly in the center of the image columns. 200 images of 56 rows and 56 pixels were taken in sequence for each object. At the operator's rate of rotation, the 200 images covered about two complete rotations of 360 degrees for each object. The capture process was intentionally not too controlled, so an object varies slightly in position and size throughout its sequence. The images were taken indoors, under florescent lighting, and an umbrella was used to reduce specularity on the objects' surfaces. The background was controlled[5] by placement of a uniform color by a sheet of gray fabric.

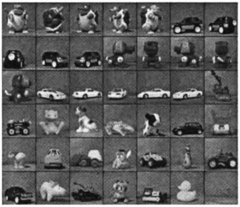


Figure 4.13: Sample(s) from each of the 25 objects classes, also showing some rotation. In the experiments, the training images were 56 × 56 and grayscale.

Including an additional "empty" (no object) class, there were $200 \times 25 + 1 = 5001$

[5] Due to automatic adjustment of the overall image intensity by the camera's capture software, later background color normalization had to be done. A program was written to do this.

images total. Every fifth image in each sequence was set aside for testing, so the other 80% were used to train the networks. Grayscale was used due to color's usefulness in class discrimination. Five top-down-enabled and five top-down-disabled networks were trained for each of the sizes: 20 × 20, 30 × 30, and 40 × 40[6]. Training involved random sample selection over 50,000 training samples. Two types of the networks were trained: the first type used excitatory top-down connections ($\beta = 0.3$), while the second type did not ($\beta = 0$).

Results

Table 4.2: Error results for MSU objects, averaged over 5 trials.

Neural plane size	Avg. error (no top-down)	Avg. error (used top-down)	Error reduction
20 × 20	8.13%	3.03%	**62.7%**
30 × 30	2.62%	0.83%	**68.3%**
40 × 40	0.63%	0.33%	**47.6%**

A neuronal population's TCG can be observed via visualization. However, for reporting purposes, TCG should be measured in some other way. The within-class scatter of neuron responses on the 2D neuronal plane can provide such a measurement.

Once a network was developed, a set of testing stimuli was used as input. The top-one neurons position was recorded. The within class scatter of firing for each stimulus class, averaged over all stimulus classes, measures how condensed the neuron responses were upon the neuronal plane. See Fig. 4.16. The class-response scatter ω is the trace of the within class scatter matrix, normalized for map size: $\omega = \sqrt{\text{tr}(\mathbf{S}_W)}/\sqrt{n}$. Using $\text{Tr}(\mathbf{A}) = \text{Tr}(\mathbf{BAB}^{-1})$, we can see this measure is invariant to rotation of the 2D map.

[6] For the larger (40 × 40 and greater) supervised networks, we scaled up in resolution to avoid neurons on the corners that didn't get updated otherwise.

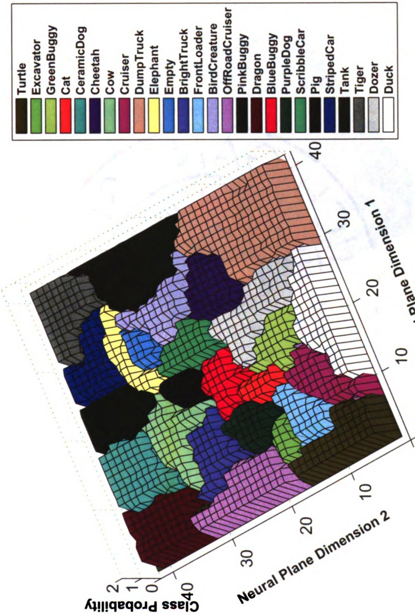


Figure 4.14: 25 Objects. Developed using top-down connections.

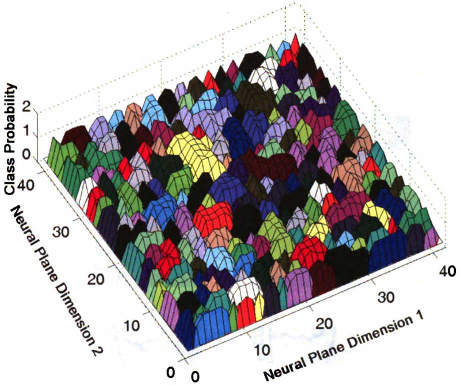
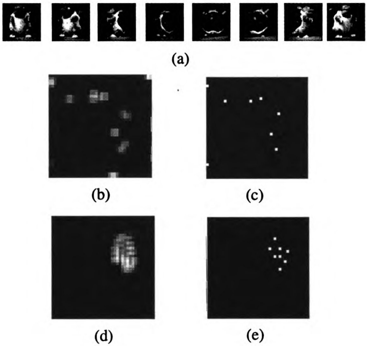


Figure 4.15: Result for 25 objects viewed from a full range of horizontal 360 degrees. Developed without using top-down connections: the maximum class firing probabilities of the 40 × 40 neurons of layer 1.

Table 4.3: Feature quality and grouping results for the experiments with 25 objects.

Top-down	Neural grid size	Developmental entropy [0-1]	Class-response scatter [0-1]
Disabled	20 × 20	0.23	0.35
Enabled		**0.19**	**0.13**
Disabled	30 × 30	0.22	0.44
Enabled		**0.15**	**0.11**
Disabled	40 × 40	0.20	0.44
Enabled		**0.08**	**0.10**



(a)

(b)

(c)

(d)

(e)

Figure 4.16: (a). Images presented to a trained network to measure the class-response scatter. (b) Bottom-up weight to the neuron representing this class ("turtle"). This network was top-down-disabled. (c) Top responding neuron positions for each of these samples for the unsupervised network (d) Layer-two weight for a top-down-enabled network (d) Top responding neuron positions for the supervised network.

Error results are presented in Table 4.2 and grouping results in Table 4.3. We tried different values of k from 1 to 10 for testing and reported the best results. The class and view angle could ideally be represented by the maps so that each neuron is responsible for a single class over a set of angles from about 5°to 25°, from largest ($n = 1600$) to smallest ($n = 400$) sizes. The results show the supervised networks develop to utilize the same amount of available resource better, shown by better error rate. Especially notable differences are seen when the number of neurons is smaller.

The per-neuron entropy is also lower, meaning the neurons are more "pure" – representing samples within a single class. The class-response scatter was significantly lower in the networks that used top-down connections. All examined top-down enabled networks showed TCG. Figures 4.14 and 4.15 show that topographic class grouping occurred in the network developed with top-down connections, but not the network without.

As a major within-class variation in this experiment, viewing angle is disregarded by the discriminant features. That is, the cortical region of each object is invariant to its viewing angle variation. It is important to note that disregarding viewing angle is not a mechanism built into the network through programming. Such an internal invariance is an emergent property of the network. Thus, other variations, such as lighting, vertical viewing angle, deformation should be applicable but further more extensive experiments are needed to quantitatively study the effects.

4.5.3 NORB Objects

The normalized-centered NORB dataset [59] is one of several 3D object recognition datasets, which is publicly available[7]. It contains binocular image pairs of five classes of objects (four legged animal, human figure, airplane, truck, car), with 10

[7] http://cs.nyu.edu/ ylclab/data/norb-v1.0/

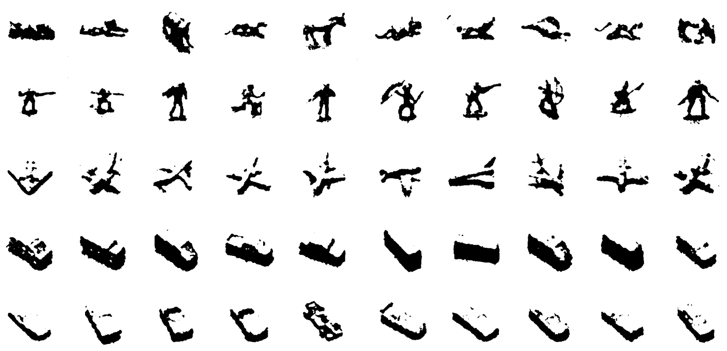


Figure 4.17: NORB objects. Each row is a different category. Examples of training samples are shown on the left, and testing (not seen in training) is shown on the right. Figure from [59].

different actual objects belonging to each class. The 5 training objects per class and 5 testing objects per class are disjoint. The small set (used here) has 24,000 training images, and 24,000 testing images, over uniform background. The dimension of each training sample is $96 \times 96 \times 2 = 18,432$. The images vary in terms of rotation (0 to 340°in 18°increments), elevation (30 to 70°in 5°increments), and lighting (6 different conditions). Recognition must be done based on shape, since all objects have roughly the same texture and color. And as the objects rotate, many of the appearances change significantly. The NORB classes are more general than those used in the 25 objects' case, so they are tougher to distinguish, but there are fewer classes. We tried top-down-disabled and top-down-enabled networks of size 40×40 and 60×60. We also tried using wraparound grids on supervised networks.

Results

Table 4.4: Error results using the normalized-centered NORB data

Method	Resource	Disjoint Test Error		
K-NN+L2 [59]	24000	18.4%		
		No TCG	TCG	Diff.
Proposed Net.	1600	26.5%	17.68%	8.82%
	3600	26.2%	**15.7%**	**10.5%**

Table 4.5: Grouping results using the NORB 5-class dataset.

Top down	Edge wrap	Neural plane size	Entropy	Scatter
N	N	40 × 40	0.5	0.41
Y	N		0.13	**0.25**
Y	Y		**0.07**	0.33
N	N	60 × 60	0.49	0.42
Y	N		0.11	**0.22**
Y	Y		**0.09**	0.33

Results are presented in Tables 4.4, which presents best results at the different sizes, and 4.5 that summarizes the grouping metrics. Figures 4.18 and 4.19 show some visualization results. A very significant difference of error of up to 10.5% is observed when comparing networks that used top-down connections to those that did not. The top-down-enabled networks exhibit purer neuron representations, lower within class scatter measures, and show TCG.

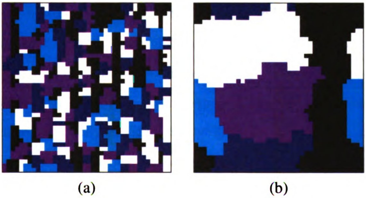


Figure 4.18: 2D class maps for a 40 × 40 neural grid after training with the NORB data. At each neuron position, a color indicates the largest outgoing weight in terms of class output. There are five classes, so there are five neurons, and five colors. (a) $\beta = 0$ (b) $\beta = 0.3$. These experiments used wraparound neighborhoods.

Using the NORB dataset allows comparison. Our method compares favorably with other methods that deal with input monolithically. See [59] for more details. With top-down connections our method outperforms K-nearest neighbor. This shows the power of within-class regularization. Nearest neighbor requires all 24,000 train-

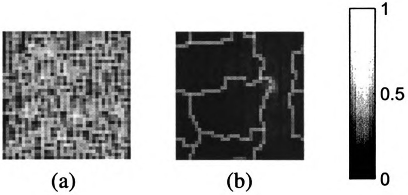


Figure 4.19: 2D entropy maps for the Fig. 4.18 experiments. High-entropy means prototypes are between class manifolds, which can lead to error. Whiter color means a higher entropy. (a) $\beta = 0$ (b) $\beta = 0.3$. Note the high entropy neurons shown here coincide with class group borders shown in Fig. 4.18.

ing samples to be stored, while the top-down-enabled networks only used $1,600$ and $3,600$ neurons. However, the network trained with disabled top-down connections showed significantly worse performance. SVM had to use significantly subsampled data (it was too slow to train with the original high dimensionality), with which is performs slightly better. The convolutional networks (6.5%) observe a significantly better error rate, but recall that convolutional networks utilize local analysis, and a fair comparison here is only with other monolithic networks. Our method is potentially more scalable than any of the other methods, due to its linear complexity, and new classes can be potentially added on the fly. Each could be learned quickly due to resource having been used for function approximation instead of for discrimination only.

4.6 Summary

The work reported here described a method in which top-down connections lead to Topographic Class Grouping (TCG). Further, the work explains why TCG leads to significantly lower error rates. Samples from different classes are far apart in the top-down-connection boosted input space, allowing class groups to grow out of smaller initial groups. The lateral excitatory pulling occurs upon class manifolds, instead of between them, reducing errors.

In the MSU 25-object data set, these networks self-organized to represent the objects over many different views using a small resource. It was shown that top-down connections led to networks that classified with lower errors and the responses for each stimulus class were grouped. With the NORB data set, it achieved a better result than K-NN using only 7% of the resource. The computational cost is linear in both the number of neurons and the input and output dimensionality in both training and testing. This method using top-down connections succeeds in object recognition and is potentially scalable for more complex real-world problems. It may contribute to a better understanding of how developmental systems can autonomously develop internal representation that is useful for desired behavior.

4.7 Adaptive Lateral Excitation

Earlier, the SOM-inspired idea of isotropic updating simulated the lateral excitatory connectivity. The updating was done in a 3×3 region around each winner neuron. The section investigates the performance effects of adaptive excitatory connections, and describes how they can lead to both smoothness and precision, in conjunction with top-down connections. The performance effects of adaptive lateral connections coupled with top-down connections have not been studied, especially in comparison to the other types of lateral excitation.

It is proposed that a major use in terms of performance of adaptive excitatory lateral connectivity and top-down connectivity is to develop abstract representative modules - statistically correlated firing groups. Modularity emerges due to the adaptivity of the local connections — as the correlations between groups decrease, the connection strengths also decrease. This serves to decrease the interference between different firing groups.

4.7.1 Motivation

Due to the pressure of evolution, the brains of organisms need to self-organize at different scales during different developmental stages. In early stages, the brain must organize globally (e.g., large cortical areas) to form "smooth" representation that is critical for superior generalization with its limited connections. At later stages, the brain must fine tune its organization for precision and modularity. Yet, lateral connectivity cannot just reduce over time (to be eventually cut off). Certain spatial structure is correlated and this correlation information should be retained. It is assumed that illusory contour detection occurs due to lateral connections, for example. But "smoothness" and "precision" are two conflicting criteria, and it seems two different organizing mechanisms are needed.

Lateral connectivity is also direction that has yet to be fully explored in terms of performance and in conjunction with these top-down connections. It alone has been handled in several different ways in various models. The self-organizing maps [55] utilize an isotropic updating function with a scheduled scope. Based on SOM, the important work of LISSOM [73] used explicitly modeled lateral connections (as weight vectors) of both excitatory and inhibitory types. The scope of the excitatory weights was smaller than the inhibitory, and the excitatory scope and learning rates adapted throughout learning. The excitatory lateral connections helped lead to organization (nearby neurons represent similar or identical features), while major effects of lateral inhibition was to encourage development of different features and to decorrelate the output – leading to a sparse response[8]. But LISSOM did not utilize a "motor" layer, and thus was not tested for performance with real-world

[8] The "winner-take-all" method used in SOM can also be considered as a form of lateral inhibition and it has the same effect – different features are developed and the output becomes sparse (an extreme case – taking the winner neuron alone to completely represent the input). Taking the "top-k" winners (neurons with the k-largest responses to the stimulus) has a similar, but more relaxed, effect.

engineering problems. We investigated [62] the use of adaptive lateral connections of excitatory type, such as those used in LISSOM.

4.7.2 Smoothness vs. Precision

Using the idea of neighborhood updating from SOM leads to a topographic LCA, where a winning neuron's neighbors are updated, with their response weighted by distance. This achieves some amount of topographic organization and cortical smoothness, depending on how the learning rate and scope of the neighborhood function are tuned.

For high-dimensional real-world data, this method has possibly introduced a new problem. In this method, the updating equation for neighbor neurons is changed to

$$\mathbf{v}_j(t) = w_1 \mathbf{v}_j(t-1) + w_2 h(n_{i,j}, t)\mathbf{x}(t-1), \tag{4.12}$$

where neuron i is a winner, and $n_{i,j}$ is the distance from neuron i's 2D position to the position of neuron j. The kernel function h defines the neighbor updating strength.

The possible issue with Eq. 4.12 is that a non-winner neuron's response y_j does *not* depend on its bottom-up and top-down combined weight vector $\mathbf{v}_j$. Instead, it is simply a function of distance from the winners. This means neurons can fire and update for stimuli that they do not represent well themselves.

This purely neighbor-based updating leads to problems for efficiently representing real data. Real world, high-dimensional data (e.g., raw pixels from a digital camera) is typically sparsely distributed – there will be large areas in the input space where a stimulus is extremely unlikely[9]. And, at least with vision, the input space tends to have multiple *disconnected* areas where stimuli are probable. Using the SOM-

[9] Consider averaging two images of two different objects. This type of "ghosting" effect (see Fig. 4.5 is not typically seen in reality.

style method of updating with any type of tuned learning rate leads to neurons representing areas of low probability between multiple high probability areas, since they are "pulled" by their neighbors closer to each of the separate high-probability areas. This phenomenon is well-documented in [20].

This phenomenon is problematic for several reasons. First, the approximation of the probability density by response is poorer since less resource is used to represent regions where the data actually lies. Second, the neurons in low probability areas do not send meaningful messages to the next layer when they fire. Since they are "between" several different high probability areas, each presumably with different meaning, their firing does not send the next level an unambiguous message. Their firing is interference between different tasks, classes, etc. This interference can lead to performance errors (depending on the data). It is also interesting to note that in biological cortex – at least in V1 – these types of between-feature averaged representations have not been observed [68].

4.8 Setup

With explicit connections, neuron i's synaptic weight vector $\mathbf{v}_i$ have the standard bottom-up (b) and top-down (e) components, but also a lateral (l) component

$$\mathbf{v}_i = \mathbf{v}_{b,i} \Downarrow \mathbf{v}_{e,i} \Downarrow \mathbf{v}_{l,i} \tag{4.13}$$

where the $\Downarrow$ symbol is for vertical vector concatenation.

In the proposed LCA with lateral excitatory connections, these connections affect the pre-competitive potential response. They take their effect *before* the winners are chosen instead of after, in the SOM-inspired method. The pre-response for a neuron is a function of its three sources from the bottom-up $\mathbf{x}(t-1)$, top-down $\mathbf{e}(t-1)$ and lateral $\mathbf{y}(t-1)$ as follows:

$$
\begin{aligned}
\hat{y}_i(t) &= g_i\Bigg(\alpha \cdot \frac{\mathbf{x}(t-1)\cdot \mathbf{v}_{b,i}(t-1)}{\|\mathbf{x}(t-1)\|\,\|\mathbf{v}_{b,i}(t-1)\|} \\
&+ \beta \cdot \frac{\mathbf{e}(t-1)\cdot \mathbf{v}_{e,i}(t-1)}{\|\mathbf{e}(t-1)\|\|\mathbf{v}_{e,i}(t-1)\|} \\
&+ \gamma \cdot \frac{\mathbf{y}(t-1)\cdot \mathbf{v}_{l,i}(t-1)}{\|\mathbf{y}(t-1)\|\|\mathbf{v}_{l,i}(t-1)\|}\Bigg) \qquad (4.14)
\end{aligned}
$$

and α, β, and γ must sum to one. They control the relative contributions of the three sources to this neuronal layer.

We update the neurons with the k largest $\hat{y}$, considered winners, using Eq. 3.13. Due to this competitive process, the developing weight vectors will only update for stimuli that they themselves represent well (unless e.g., α is too small). It will be less likely to have damaging interpolation as is seen in the 3×3 updating case when using this method. The interpolation provided by lateral connectivity should now be useful (within high-probability areas).

4.8.1 Initialization

– How are the lateral excitatory weights to be initialized? Similar to in LISSOM, and based on observations that most lateral excitatory connections are short-range [17], the scope of connectivity is restricted. For example, a neuron can only excite another neuron up to 5 neural positions away. And the actual values of the weights within the scope of connectivity are determined by an isotropic function such as a Gaussian. These initial weights will organize cortical representation to help pull similar features together in the physical neuron map. However, we also want to adapt the weights so that features with low correlation can exist nearby without interfering with one another – the lateral weight between them will diminish and be cut.

4.8.2 Adaptation

– Biological lateral connections are strong between functionally similar neurons [72, 110]. While the purpose of lateral excitatory weights early in development is to drive topographic organization, the purpose of later *adaptation* in lateral excitatory weights is to develop weights between nearby neurons that reflect the correlation of firing of those neurons. We can use LCA's updating equation exactly for lateral weights and we will develop a weight between neuron i (which is a winner and has nonzero firing rate) and j (which is within the scope defined by m but may or may not be a winner) equal to

$$E(y_i(t)y_j(t-1)|y_i(t) > 0). \tag{4.15}$$

which is the expectation of firing rate of neuron j when neuron i has fired, weighted by neuron i's own firing rates. This is simply Hebb's principle applied to the lateral weight.

4.8.3 Developmental Scheduling

– As is necessary for in-place learning, each neuron has a self-stored age, and an age-dependent updating schedule that defines w_1 and w_2. However, adaptation of the lateral weights must be scheduled differently from the bottom-up and top-down weights. This is due to a simultaneous dependency in development – the bottom-up and top-down weights depend on the lateral connections early on for their development and topographic organization, and the lateral connections cannot reliably begin to reflect Eq. 4.15 until this organization has settled – meaning the adaptation of bottom-up and top-down weights has settled down. Therefore, the lateral connections must have more plasticity, later, than the bottom-up and top-down connections.

How can a single neuron have two separate updating schedules when it does not know the origin of its synapses? We can consider the lateral connections to be representative of a different cell type which we do not directly model – interneurons dealing with one-to-one connectivity in the same layer. This cell type could perhaps then have a different schedule of plasticity.

The performance effect of Eq. 3.13 is to cut off connections between areas of the stimulus space that do not correlate, thereby avoiding the problems that come from neuronal neighbor pulling. This lateral (and top-down) updating leads to more *modular* collections of neurons – functionally related neuron groups with lower between-group interference than compared to the 3 × 3 method. And, due to the lower interference, each feature will tend to represent the higher-probability areas (where the data is actually observed).

4.9 Experiments

We compared the laterally connected LCA algorithm with an LCA method without lateral connections ($\beta = 0$), that instead used 3×3 updating. The data was the MSU-25 Objects dataset. All tests utilized a neuronal plane size of 20×20 neurons.They connect to a motor layer of 26 neurons, which had top-down projections back to the feature layer. In testing, the highest responding motor neuron was taken as the guessed class.

We used the schedule of inhibition strength, or connection scope (changing k), shown in Table 4.6 and did not allow the lateral connections (if they were enabled) to adapt until $t = 500$. Each sample input was repeated for five iterations, and we did not update synapses until after the last iteration per sample. This allowed the lateral activity to settle. In training, the correct motor output was imposed. The response was reset for each new training or testing sample, since such temporally discontinuous experience (jumping directly to a new image with a new object class

Table 4.6: Scheduling of inhibitory scope.

Sample number	Number of winners (k)
0	20
1000	15
2000	5
3000	3
4000	1

having no transition) is not experienced in reality.

Results are shown in figures 4.20 and 4.21. The version with adaptive lateral connections and adaptive top-down connections ($\alpha = \beta = \gamma = 0.33$) is the best. It is interesting that the laterally connected version *without* using top-down ($\alpha = \gamma = 0.5$) matches the 3×3 version *with* top-down. Why is this the case? Fig. 4.21 shows the mean per-neuron class-entropy (see Appendix). If this is high, the neurons are firing for more than one class. We claimed this type of interference leads to errors. Indeed the two versions using 3×3 updating have significantly greater entropy on average than the ones using the lateral connections. The effects of the two methods can be visualized by looking at the bottom-up weights as images, as in Fig. 4.22.

4.9.1 Comparison

We also compared with an LCA algorithm with lateral connections but instead using the "dot-product" SOM updating equation [55]

$$\mathbf{v}_i(t) = \frac{\mathbf{v}_i(t-1) + \eta(t)\mathbf{x}(t-1)}{\|\mathbf{v}_i(t-1) + \eta(t)\mathbf{x}(t-1)\|_2} \tag{4.16}$$

where $\mathbf{v}_i$ is the winning component vector at time t.

We also compared by instead using the LISSOM updating equation [73][10]

[10] Note that we are not comparing with SOM and LISSOM – we simply replace LCA's updating equation

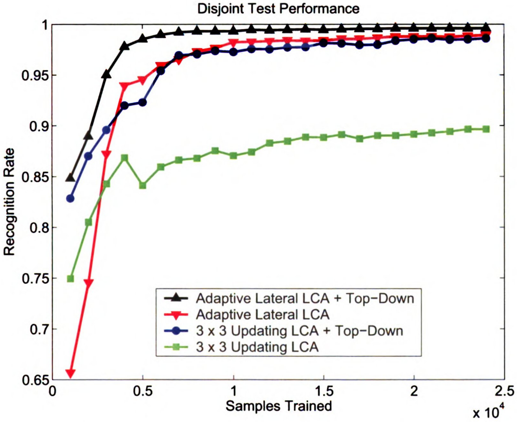


Figure 4.20: Performance comparison of LCA using 3×3 updating to LCA with excitatory lateral connections with and without top-down connections.

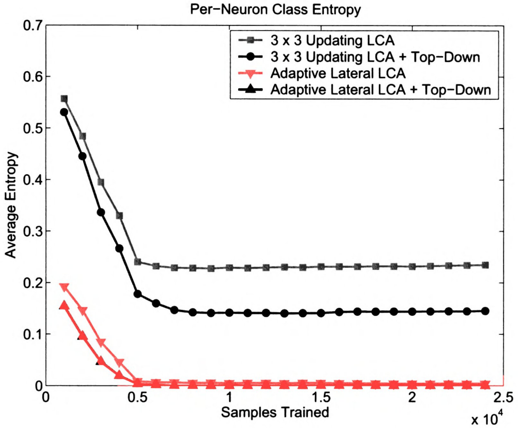


Figure 4.21: Per-neuron class entropy for the four variants of LCA in the tests with the 25-Objects data.

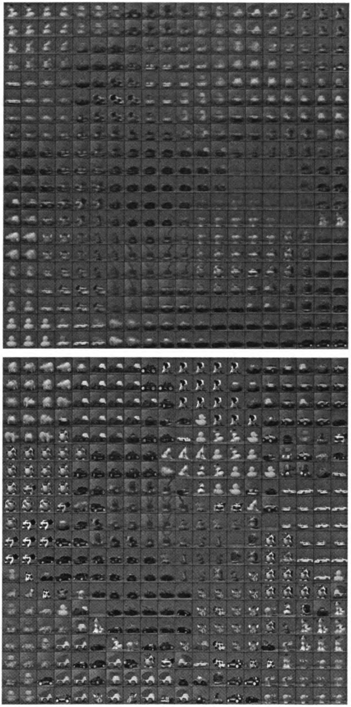


Figure 4.22: Above is the bottom-up weights after development for a neural map (each weight can be viewed as an image) that utilized 3 × 3 updating without top-down. Below are the weights for a neural map that developed using explicit adaptive lateral connections (also without top-down). Note the smearing of the features above and the relatively higher precision of representation below, while still being somewhat topographically organized.

$$\mathbf{v}_i(t) = \frac{\mathbf{v}_i(t-1) + \eta(t)\mathbf{x}(t-1)y_i(t)}{\|\mathbf{v}_i(t-1) + \eta(t)\mathbf{x}(t-1)y_i(t)\|_1} \tag{4.17}$$

The tuning of $\eta(t)$ for both of these is in general not simple [11]. Since the two above updating methods use only a single learning rate parameter to adapt the neuron weights to each new updating input, and a method to bound the strengths of synaptic efficacies (e.g., vector normalization), while CCI LCA uses the time-varying retention rate $w_1(t)$ and learning rate $w_2(t)$, where $w_1(t) + w_2(t) = 1$, in order to *optimally* maintain the energy estimate (as formalized in Section ??, and in order to achieve optimal representation. With the energy gone in the SOM and LISSOM updating schemes above, there is no way to adjust the learning rate $\eta(t)$ to be equivalent to the optimal LCA scheduling.

The result in Fig. 4.23 supports this, as the non-LCA updating methods led to much worse performance. Interestingly, the entropy and organization of the feature layers was not significantly different between the three methods. The problem arose for the motor neuron's bottom-up weights, which were not optimal and unstable for the SOM and LISSOM updating methods.

4.10 Summary

Efficient (not wasting the available resource) and effective (leading to good performance) emergent internal representation is crucial for development. In published computational cortical maps, self-organization – topographic "smoothness" – is often achieved at the cost of high precision. The work reported here showed adaptive lateral excitatory connections used developmental scheduling to both self-organize a cortical map and to develop feature subgroups without cross-group interference that

[11] We used the "power" equation with initial learning rate of 0.1 for the SOM method [114], and based our tuning of the LISSOM equation from the appendix in [73]

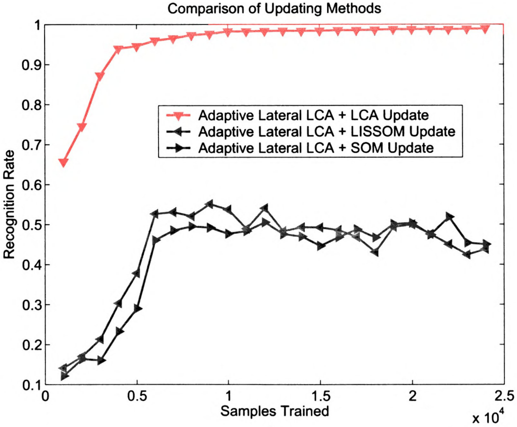


Figure 4.23: Performance comparison for different updating methods, using lateral connections without top-down.

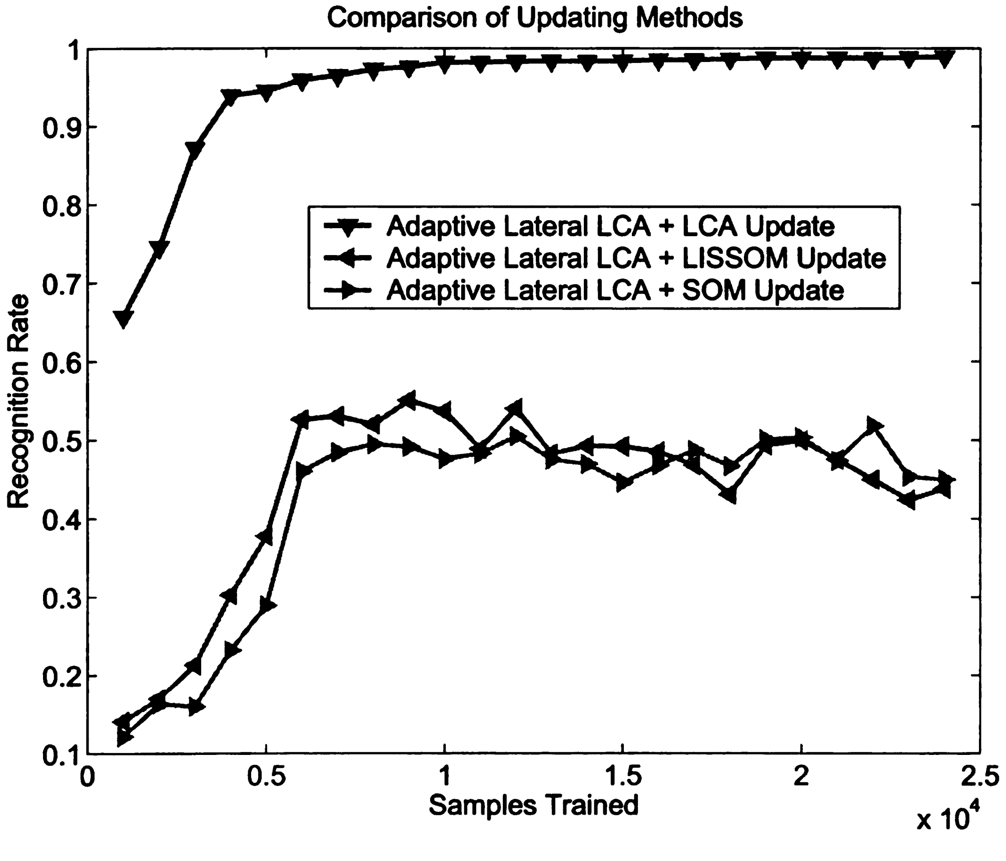


Figure 4.23: Performance comparison for different updating methods, using lateral connections without top-down.

traditional (e.g., 3 × 3 updating) methods exhibit. The performance improvements are effects of several cortex inspired mechanisms including top-down connections, dually optimal LCA updating, and local sorting that simulated lateral inhibition without requiring many iterations. Under these mechanisms, it was shown that global-to-local scope scheduling and adaptive lateral connections leads to effective and efficient self-organization.

4.11 Bibliographical Notes

Neural networks have traditionally operated using bottom-up (feed-forward) connections as primary connections, with the top-down (feedback) connections not used at all or approximately used in a constrained weight-tuning mode. For example, backpropagation-based networks [58, 126] use the top-down *error signal* to train the bottom-up weights, but it does not use explicit top-down connections. A few networks used top-down information as part the input. The use of an expanded input including both input and output vectors for the Self-Organizing Maps (SOM) was briefly mentioned as a possibility by Kohonen 1997 [54], and was also used as an input to a Hierarchical Discriminant Regressor (HDR) [129]. In the laterally connected self-organizing LISSOM [93] and the Multi-layer In-place Learning Network (MILN) [120], neurons take input from bottom-up, lateral and top-down connections. For LISSOM, Sit & Miikkulainen 2006 [93] explained how a neuron responding to an edge, for example, can develop to receive top-down feedback from neurons that detect corners including that edge in the next layer. LISSOM did not utilize an output layer, however.

4.12 Appendix: Neuronal Entropy

Each neurons "purity" with respect to class can be measured by using entropy. Developmental entropy takes into account the updating history. It is generally correlated with firing entropy of a mature neuron. Neuron i developed in a pure way if its entropy of stimulus history is small. From Eq. (4.8), it can be seen that bottom-up weight $\mathbf{w}_{b,i}$ of neuron i is a weighted sum of input samples which have been used in updating:

A neuron with low developmental entropy was updated with most samples from the same class. A neuron with high entropy was updated by samples from many different classes. First, measure per-neuron probability with respect to each class.

$$\mathbf{w}_{b,i} = \sum_{t=1}^{m_i} \alpha_{2,i}(t)\mathbf{b}_i(t) \tag{4.18}$$

where $\mathbf{b}_i(t)$ are stimuli (samples) used to update the neuron $\mathbf{w}_{b,i}$ and $\alpha_{2,i}(t)$ are the corresponding update weights. To measure the purity of each neuron, we define the empirical "probability" that samples arose from class j as:

$$p_j = \frac{\sum_d \alpha_{2,i}(j)}{\sum_{t=1}^{m_j} \alpha_{2,i}(t)} \qquad d \in \text{class } j \tag{4.19}$$

Let the matrix $\mathbf{P} = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$ be the matrix of probabilities for n neurons where there is a distribution $\mathbf{p}_i = \{p_{i,1}, p_{i,2}, ..., p_{i,c}\}$ for each neuron. To quantify the "purity" of the probability distribution for the i-th neuron, we have

$$\varepsilon_i = -\sum_{d=1}^{c} p_{i,d} \log_c(p_{i,d}) \tag{4.20}$$

If $\varepsilon_i = 1$, then neuron i was updated using inputs arising from a single class.

Chapter 5

Recurrent Dynamics and Modularity

About 85% of cortical neurons are excitatory, and about 85% of their synapses are to other excitatory neurons [29]. This suggests that excitatory feedback circuits are very prevalent throughout cortex, yet computational models and neural networks have not typically utilized excitatory feedback.

Many actions are not reflexive response to stimuli, but are instead a result of deliberation over some time. Such deliberation may be purely internal, but also could utilize a constant stimuli stream from the environment. This chapter considers the second case. This chapter presents theory, experiments and results pertaining to computational multilayer Hebbian neural networks that use both bottom-up and top-down connections as dynamic systems in time. The top-down connections provide *temporal context* — a "bias" or "expectation" of the next sensation based on previous sensation. Using this temporal context, the performance in a challenging object recognition task is shown to becomes nearly perfect on each view when the data is sequential (video). If we assume the agent is not required to answer regarding the identity of the object within the first few frames after a transition from looking at one object to another, the performance becomes 100%.

In previous work, top-down connections were shown to cause a motor-initiated biasing effect during development. This led to a biased compression of internal representation so that the relevant information about the input was prioritized over the irrelevant information, leading to better performance as compared to networks that did not use top-down connections [64].

Here, I'll examine the effect of the meaning-carrying top-down connections on the lower layer of neurons over time. We assume the network has already undergone a significant amount of learning. The class-conditional entropy of the feature neurons, as reflected in the top-down connection strengths, has a major impact. Three network types are analyzed, in a linear approximation of the actual network, which is nonlinear due to the lateral inhibitory mechanism used. A minimum entropy network can also be interpreted as modular as every feature neuron is associated with only one motor neuron. In such a case, the recurrent activity effect spreads activation appropriately to all the associated features of each active motor neuron. A high entropy network has widespread connectivity. There is a single steady-state distribution of activity, which contains nonzero activation for all neurons. Given any initial input, this type of network will always converge to the same pattern of activation. A low entropy network is nearly modular, but has a few shared between-module connections. It has the same properties as the high-entropy network, but lateral inhibition is proposed as a mechanism to functionally convert a low-entropy network into a minimum entropy network, and thus control the effect of top-down excitation.

5.1 Motivation

Intuitively, stability of perception is a core difference for an artificial network operating in the real world as compared to one operating on stored data. In realistic environment, an object tends not to blink in and out of locations, or change how it looks all of a sudden. The changes in location and appearance tend to be gradual. However, there can be sudden changes. This spatial and temporal locality is taken advantage of in biological networks, and it should also be taken advantage of in artificial networks.

Humans use temporal context to guide visual recognition. Normal human adults do not experience the world as a disjointed set of moments. Instead, each moment contributes to the context by which the next is evaluated. Is our ability to utilize temporal context in decision making innate or developed? There is evidence that the ability may be developed from a child's genetic programming so that it emerges several months after birth. From much experimental evidence, Piaget [84] stated that before they are around 10 months old, children do not visually experience objects in sequences, but instead as disassociated images. Many of Piaget's tests measured overt behavior and could not measure internal decisions, however. Baillargeon [3] later found evidence of object permanence, an awareness of object existence even when the objects are hidden from view, in children as young as 3.5 months. It illustrated an aspect of the covert (internal) process behind recognition. Evidence of this awareness in significantly younger infants is not supported. How does this ability to predict emerge? Does it emerge from interactions with the environment (i.e., would a child in an environment obeying different laws of physics learn to predict differently?) or is it genetically programmed to emerge at a specific time?

It seems that after sufficient developmental experience, children will generate an internal expectation from recent experience that is used for biasing recognition. The network basis for generating this expectation is not clear. Object permanence, specifically the drawbridge experiment, may be a special case that can be solved through a set of "occlusion detectors", such as those found in the superior central sulcus in the ventral visual pathway [4]. How the brain creates prediction signals in general relates to the fundamental question of how the brain represents time.

Buonomano [16] discussed the two prevalent views of how this may be – "labeled lines", in which each neuron's firing is explicitly representative of a certain amount of time, or "population clocks", where the temporal information is represented by the overall population dynamics of local neural circuits. In the latter, each individual neuron carries no timing information. Such local circuits would be highly recurrent, and have connections from other areas, where external events arise and perturb the circuit's state.

There are many bottom-up connections from inferior temporal cortex (ITC) to prefrontal cortex (PFC) and many top-down connections from PFC to ITC. It is hypothesized that neurons in PFC perform behaviorally-relevant category binding, while neurons in ITC are responsive to high-level visual features [35]. Therefore it is thought the bottom-up connections provide information about detected high-level features to PFC, which binds them together for categorization. But all the computational roles of the top-down connections are currently not known, specifically in development and learning.

We seek to verify two general ideas about top-down connections via simulation: that they could act as the impetus of category-specific self-organization, e.g., seen in the fusiform face area (FFA) and parahippocampal place area (PPA), and that they can act as a "bias" (memory store) for biasing the ITC features [104]. We built networks with three interconnected neuronal layers: a sensory area (layer one), a feature representation area (layer-two: ITC), and a category-behavior area (layer-three: PFC). In the network presented here, each layer-two neuron receives excitatory inputs from the bottom-up, laterally, and top-down and each layer-three neuron has bottom-up inputs. Neurons compete with others on the same layer through lateral inhibition. Neurons that are not firing-inhibited learn through a Hebbian learning algorithm, in which the strength of synaptic learning is based on presynaptic and postsynaptic potentials. In contrast to slow feature analysis methods [34], our network's development does not depend on slowly changing inputs, but instead the correlations between the category information from PFC and the true class of the stimulus.

5.2 Concepts and Theory

The same mechanism of motor-boosted distance we discussed in the last chapter can be used to make use of this temporal context. Here, the network generates the top-down context on its own, over a set of input frames. If we use a nonzero top-down parameter in the testing phase, we create a temporally sensitive network for use over realistic video streams.

Once a network has developed through supervision, it can run without supervision to classify stimuli. Top-down connections can play a significant role after development. When the input is temporally continuous (e.g., video sequences), the top-down connections can provide temporal context. The classification at any time step will feed back to bias feature neurons and will affect the sensation at the next time step.

5.2.1 Internal Expectation

For realistic video data, a spatiotemporal decision is expected to be more accurate than a merely spatial decision. We are given a sample $\mathbf{x}(t)$ that has not been seen before from one class in $C = (c_1, c_2, ..., c_d)$. Assume we have accurate *a posteriori* estimates of class $p(c_i|\mathbf{x}(t))$, then taking the maximum over C as the result will give the optimal Bayesian choice. But depending on the spatial distribution, the error rate of this spatially optimal choice could still be unsuitably high (see Fig. 5.1).

If there's temporal locality in the data (e.g., there's a 90% chance the class of a sample at $t + 1$ is the same as the class of the sample at t), using *a posteriori*

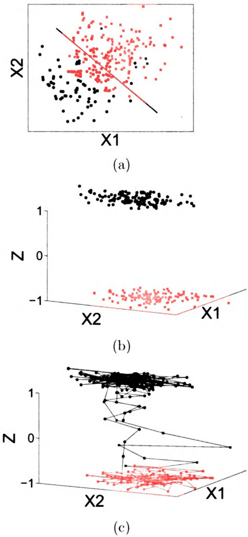


Figure 5.1: Weakly separable data can become more strongly separable using temporal information. (a) Two classes of data drawn from overlapping Gaussian distributions in space. (b) Adding another dimension: perfect separation trivially occurs if one includes another dimension which depends on the label of each point, but of course the label is not available. (c) When the data has much temporal continuity, the previous guess of class can be used as the third dimension. **z** becomes an expectation by using the guessed label of the current point and the previous **z**. Temporal trajectories are shown here. The points in the middle are "transition points", where expectation is not as strong since recent data was from the other class.

probabilities over a spatiotemporal window of k frames: $p(c_i|\mathbf{x}(t), \mathbf{x}(t-1), ..., \mathbf{x}(t-k))$ lead to a much lower error rate. First, it is unknown what k to select. If an event important to the current decision is made more than k frames in the past, it will be forgotten. Practically, due to the high-dimensionality by using the raw input vectors (i.e., high-dimensional images), this is very tough to estimate for even a moderately large k.

If instead we keep a single vector parameter ($\mathbf{z}$), which is updated incrementally by some function f: $\mathbf{z}(t) = f(\mathbf{x}(t-1), \mathbf{z}(t-1))$, then the problem becomes estimation of $p(c_i|\mathbf{x}(t), \mathbf{z}(t))$. A first advantage over the last form is that $\mathbf{z}$ potentially can store information from as far back in time as possible. A second major advantage is the potential of $\mathbf{z}$ to be compressed in a more abstract form than $\mathbf{x}$. For example, if the recent image contained a cat, it can be compressed as small as a single neuron activation (representing the abstract "cat"). This removes all the *irrelevant* information from the past and stores only the *relevant* information.

By f, old data gets integrated into the current state, thereby "hashing" temporal information into spatial. In the network presented here, motor output $\mathbf{z}$ acts as an abstract memory, which is updated after each sample. It feeds back to the feature layer as a "prior" that biases the next decision.

5.2.2 Network Overview

The three-layer network to consider is shown in Fig. 5.2. The sensory (input) layer's activation is represented by $\mathbf{x}$, the hidden (feature) layer's activation is represented by $\mathbf{y}$ and the motor (output) layer's activation is represented by $\mathbf{z}$. Sensitivities (weights) of the feature neurons to the input are column vectors of matrix $\mathbf{V}$. Top-down weights of feature neurons from motor neurons are column vectors of matrix $\mathbf{M}$. Bottom-up weights of motor neurons from feature neurons are column vectors of matrix $\mathbf{W}$. There are n feature neurons and c motor neurons. The motor neurons

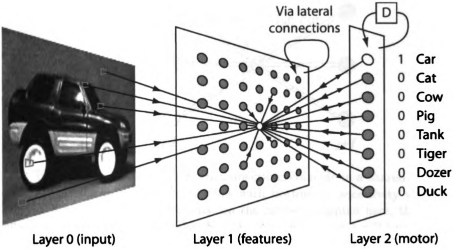


Figure 5.2: A three-layer network structure. The internal layer 1 takes three types of input: bottom-up input from layer 0, top-down input from layer 2 and lateral input from the neurons in the same layer. The top-down input is considered delayed as it is the layer 2 firing from the last time step. The "D" module represents this delay. The connectivity of the three connection types is global, meaning each neuron gets input from every other neuron on the lower and higher layers. Lateral inhibition is handled in an approximate sense via k-winners take all. Figure courtesy of Juyang Weng.

are called such since they control action. We utilize a hardcoded action production module, which simply uses the index of the motor neuron with the largest firing rate: $\arg\max_{1 \leq i \leq c}\{z_i\}$. It maps the label to a word, that was used in training to teach the class.

Compose **W** and **M** so that bottom-up and top-down weights are linked, but not exactly shared (see Fig. 5.3). We also want columns of **W** and **M** to have unit l1-norm. So, let $\hat{\mathbf{W}} = \hat{\mathbf{M}}^T$, where $\hat{\mathbf{W}}$ are the non-normalized bottom-up motor weights. Then columns of **W** and **M** are l1-normed from $\hat{\mathbf{W}}$ or $\hat{\mathbf{M}}$. By using l1-norm, the feature-motor weights have a probabilistic interpretation. For a motor neuron i, its bottom-up weight from feature neuron j represents $p(y_i^m(t) > 0 | y_j^f(t-1) > 0)$ — the probability the motor fires if the motor neuron fired last. For a feature neuron i, its top-down weight from motor neuron j represents $p(y_i^f(t) > 0 | y_j^m(t-1) > 0)$.

The feature layer's firing rate vector **y** is a function of bottom-up activity **x** and top-down firing rates **z**:

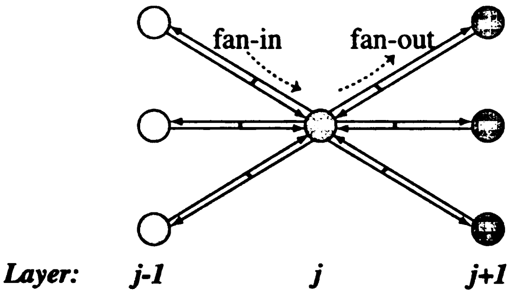


Figure 5.3: How bottom-up and top-down connections coincide in a shared way. Looking at one neuron, the fan-in weight vector deals with bottom-up sensitivity while the fan-out weight deals with top-down sensitivity. In the model presented here, the weights are shared among each two-way weight pairs. So, a neuron on layer $j+1$ will have a bottom-up weight that is linked to the top-down weight of a neuron on layer j.

$$\mathbf{y}(t) = f_{LC}(\mathbf{x}(t), \mathbf{z}(t)|\mathbf{V}, \mathbf{M}). \tag{5.1}$$

where f_{LC} denotes the per-layer computation algorithm. The motor layer uses the firing of the feature layer as its input:

$$\mathbf{z}(t+1) = f_{LC}(\mathbf{y}(t), \mathbf{0}|\mathbf{W}, \mathbf{0}). \tag{5.2}$$

The recurrence occurs since the firing of the motor layer is fed back to the feature layer. For purposes of analysis, let all network weights be frozen (no learning).

5.2.3 Network Dynamics

For purpose of analysis, we'll first examine a simpler linear system. Then, we'll discuss the effect of adding in the nonlinear lateral inhibition and other effects.

For the linear system, let the layer computation function $c = f_{LC}(\mathbf{a}, \mathbf{b}|\mathbf{A}, \mathbf{B})$ combine the bottom-up and top-down activity as follows:

$$\mathbf{c} = f_{LC}(\mathbf{a}, \mathbf{b}, |\mathbf{A}, \mathbf{B}) = (1-\alpha)\mathbf{a}\ \mathbf{A} + \alpha\mathbf{b}\ \mathbf{B} \tag{5.3}$$

Then,

$$\begin{aligned} \mathbf{y}(t) &= (1-\alpha)\ \mathbf{x}(t)\ \mathbf{V} + \alpha \mathbf{z}(t)\ \mathbf{M} \qquad (5.4) \\ \mathbf{z}(t+1) &= \mathbf{y}(t)\ \mathbf{W} \qquad (5.5) \end{aligned}$$

and by using $\mathbf{z}(t) = \mathbf{y}(t-1)\ \mathbf{W}$, and substituting for $\mathbf{z}$, we get

$$\mathbf{y}(t) = (1-\alpha)\mathbf{x}(t)\ \mathbf{V} + \alpha \mathbf{y}(t-1)\ \mathbf{W}\ \mathbf{M} \qquad (5.6)$$

The matrix $\hat{\mathbf{A}} = \mathbf{W}\ \mathbf{M}$ contains all the recurrence in the system. Since both $\mathbf{W}$ and $\mathbf{M}$ are column stochastic matrices, $\hat{\mathbf{A}}$ is column stochastic. Any element $\hat{A}_{i,j}$ represents the indirect flow of excitation from neuron i to j[1]: $\hat{A}_{i,j} = \mathit{flow}(i,j)$, where

$$\mathit{flow}(i,j) = \sum_{k=1}^{c} w_{i,k}\ m_{k,j} \qquad (5.7)$$

For the formulation of this system, $\mathit{flow}(i,j) = \mathit{flow}(j,i)$, and therefore matrix $\hat{\mathbf{A}}$ is also symmetric and row stochastic. A system with this conservation of flow property is inherently stable. The purpose of this work is to examine the distribution of excitation over time without concern for issues of stability.

A stochastic matrix defines a Markov chain. $\hat{\mathbf{A}}$ models transition probabilities between feature neurons if only one feature neuron is active at any time; it also can describe the percent of excitation routed from any active feature neuron f_i to the neurons it connect to. These transitions are not direct, but instead go through the motor neurons. Similarly, the matrix $\mathbf{W}^T\ \mathbf{M}^T$ is an excitation routing matrix for motor neurons.

To put the above into the standard form for a linear time-invariant system, let

[1] There is no direct flow here, as all feature-feature connectivity is through the motor neurons.

$\mathbf{A} = \alpha \hat{\mathbf{M}}^T \hat{\mathbf{W}}^T$, $\mathbf{B} = (1-\alpha)I$, and $\mathbf{u}(t) = \mathbf{V}^T \mathbf{x}^T(t)$:

$$\mathbf{y}^T(t+1) = \mathbf{A}\mathbf{y}^T(t) + \mathbf{B}\mathbf{u}(t). \tag{5.8}$$

(For further analysis, assume $\mathbf{y}$ means $\mathbf{y}^T$.)

The closed form solution to the above equation [1] is:

$$\mathbf{y}(k) = \mathbf{A}^k \mathbf{y}(0) + \sum_{j=0}^{k-1} \mathbf{A}^{k-j-1} \, \mathbf{B} \, \mathbf{u}(j). \tag{5.9}$$

We'll use eq. 5.9 to analyze the network's behavior over time for different types of connectivity described by the excitation routing matrix.

Due to positive feedback, this system is generally characterized by spread of activity. We wish to show the usefulness of top-down connections if the connectivity described in the recurrence matrix is *modular*. Otherwise, problems from the unchecked positive feedback occur when the features are not highly selective (non minimal entropy) and the connectivity is widespread and nonmodular. Encouragingly, the spread of positive feedback in low entropy systems is potentially manageable, while it is not in high entropy systems.

5.2.4 Minimum-Entropy Networks

Let a network with every neuron having zero entropy be called *strictly modular*. In a strictly modular network, each feature neuron is only associated with a single motor neuron, as its connections to the other motors will equal zero (see Fig. ??)(a). Each motor and its associated features is called a module. Viewed as a graph, it would consist of m disconnected components.

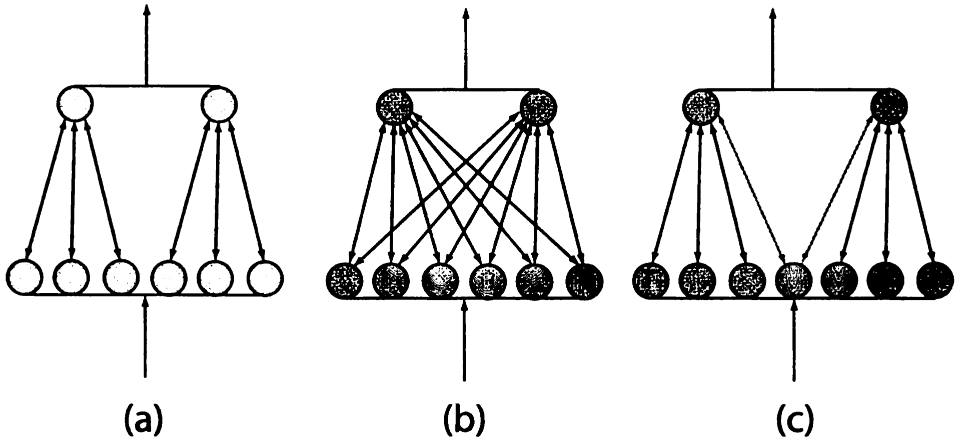


Figure 5.4: Examples of the three types of connectivity. (a) Minimum entropy network. Top-down feedback has a sensible and reliable effect of biasing the features associated with the firing motors at a level appropriate to the motors' firing. (b) High entropy network. Top-down feedback is not useful in this type of network since it spreads quickly. (c) Modular network. Here, the neurons are minimum entropy except for one "border" neuron. Unchecked, positive feedback spreads throughout, but this situation is manageable through lateral inhibition, in which the low activity connection is inhibited, and the network acts as a minimum entropy network.

Internal Activity Only

If the external (sensory) input is removed, what is this network's behavior when there is some internal firing? We will show here that each firing motor potentiates only its associated features evenly, and all activity eventually dies out. Let $\mathbf{u}(t) = \mathbf{0}$, but $\mathbf{y}(0) \neq \mathbf{0}$, meaning there is some existing activity stored as current context, but the sensory input has turned off (i.e., the eyes are closed). Eq. 5.9 becomes $\mathbf{y}(k) = \mathbf{A}^k\mathbf{y}(0) = \alpha^k\hat{\mathbf{A}}^k$, where $\hat{\mathbf{A}} = \mathbf{M}^T\mathbf{W}^T$.

For $\hat{\mathbf{A}}$, each of the matrices $\mathbf{W}$ and $\mathbf{M}$ are column stochastic. Therefore $\hat{\mathbf{A}}$ is also column stochastic. It describes a markov chain over the feature neurons, linked to each other indirectly through motor neurons. A strictly modular network with multiple motors is actually a *reducible* markov chain, with m sets of closed states (modules) and no transition states. Each module corresponds to a mode of operation. Without external input, depending on the existing internal context $\mathbf{y}(0)$, it can “enter” any of the modules, but activity cannot spread between modules.

The feature excitation routing matrix of this type of markov chain has the form

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{C}_1 & \mathbf{0} & ... & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_2 & ... & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & ... & \mathbf{C}_m \end{bmatrix}$$

. Each internal submatrix is square, column stochastic and strictly positive. They describe the routing activity within the different firing modules.

Lemma 5.2.1. *For a strictly modular network,* $\hat{\mathbf{A}}^k = \hat{\mathbf{A}}$.

Proof. Any module’s internal routing matrix $\mathbf{C}_i$ can be decomposed into pairs of r eigenvalues and associated matrices [37] as $\mathbf{C}_i^k = \lambda_{1,i}^k \mathbf{C}_i^{(1)} + \lambda_{2,i}^k \mathbf{C}_i^{(2)} + ... + \lambda_{r,i}^k \mathbf{C}_i^{(r)}$. Each submatrix in $\hat{\mathbf{A}}$ is rank one, and thus has only one eigenvalue, which will be equal to one. Therefore, for any module i, $\hat{\mathbf{C}_i}^k = \mathbf{C}_i$. It follows that $\hat{\mathbf{A}}^k = \hat{\mathbf{A}}$. □

Case 1: $\alpha = 1$: Since $\hat{\mathbf{A}}^k = \hat{\mathbf{A}}$, the distribution of excitation at any future frame is the same as the distribution at just the next frame. Thus, each firing motor neuron will distribute its firing equally (since its nonzero top down weights each equal one due to L1 norm) among its associated features. Those features’ firing feed back into the same motor neuron. There will be no spread of excitation among different modules.

Case 2: $0 \leq \alpha < 1$: The network operates as described above but all internal activity will decrease exponentially over time due to α^k.

The number of steady state distributions of $\hat{\mathbf{A}}$ is the number of modules, since the eigenvalues of a disconnected graph are the eigenvalues of its connected components. If the features initially firing are all linked to the same motor (single mode case), the network will stay within that motor’s mode of operation. Activity cannot spread to another module. When operating in one mode, the features associated with that motor are biased and the features not associated with that motor are not biased.

When multiple motors fire, the excitation is distributed based on $\mathbf{A}$, but there is no sharing between modules. Let any motor i have firing rate z_i. Since each motor distributes firing potential evenly among its features, the potentiation of each of its n_i features is z_i/n_i.

Internal Activity and External Stimulation

Even with such controlled feedback, the usefulness of the top-down connections in the above case in general seems to require

$$E[\mathbf{z}_i(t)|\mathbf{x}(t-1) \in C_i] > E[\mathbf{z}_j(t)|\mathbf{x}(t-1) \in C_i], \ \forall i \neq j \tag{5.10}$$

where C_i is the class label represented by motor i. The truth of the above conjecture depends on 1. the spatial probability distributions of each class, 2. the amount of spatial overlap of these distributions, 3. the features used in $\mathbf{V}$ to represent input space $\mathcal{X}$, and 4. the amount of temporal locality of class label in the data. If conditions are such that this expectation difference exists for data that is not perfectly classifiable without time and the samples are drawn spatially i.i.d., then the added $\mathbf{z}$ dimensions should make the optimal spatiotemporal decision boundary better than the optimal spatial boundary (refer to Fig. 5.1).

Class Transitions: Avoiding Hallucination

A key issue is that of transitions from experiencing stimuli from input class to another. The internal context will be inaccurate at these points. As shown here, a strictly modular network will recover. Recall Eq. 5.9. We set time 0 to the transition point. Assume that the internal context $\mathbf{y}(0)$ strongly biases one class, but all the next input frames are of another class. For simplicity let them be the same image so that: $\mathbf{u}(j) = \mathbf{u}(0), \ \forall j$.

In a strictly modular network with an arbitrary internal context $\mathbf{y}(0)$ given the same input $\mathbf{x}$ over time, motor activity $\mathbf{z}(k)$ approaches $\mathbf{W}^T\mathbf{V}^T\mathbf{x}$.

Proof. Using Eq. 5.9 and Lemma 5.2.1:

$$\begin{aligned}
\mathbf{y}(k) &= \alpha^k\hat{\mathbf{A}} \\
&+ \hat{\mathbf{A}}\alpha^{k-1}(1-\alpha)\mathbf{u}(0) \\
&+ \hat{\mathbf{A}}\alpha^{k-2}(1-\alpha)\mathbf{u}(0) \\
&+ \ldots \\
&+ \hat{\mathbf{A}}\alpha(1-\alpha)\mathbf{u}(0) \\
&+ (1-\alpha)\mathbf{u}(0) \\
&= \alpha^k\hat{\mathbf{A}} - \alpha^k\hat{\mathbf{A}}\mathbf{u}(0) + \alpha(\hat{\mathbf{A}}\mathbf{u}(0) - \mathbf{u}(0)) + \mathbf{u}(0). \qquad (5.11)
\end{aligned}$$

As k increases with $0 \leq \alpha < 1$, this approaches $\alpha(\hat{\mathbf{A}}\mathbf{u}(0) - \mathbf{u}(0)) + \mathbf{u}(0)$.

To get motor output $\mathbf{z}(k)$, we project onto $\hat{\mathbf{W}}$. For a strictly modular network, the motor transition matrix $\hat{\mathbf{W}}^T\hat{\mathbf{M}}^T = \mathbf{I}$. Thus, $\hat{\mathbf{W}}^T(\alpha\hat{\mathbf{A}}\mathbf{u}(0) - \alpha\mathbf{u}(0) + \mathbf{u}(0)) = \alpha\hat{\mathbf{W}}^T\mathbf{u}(0) - \alpha\hat{\mathbf{W}}^T\mathbf{u}(0) + \hat{\mathbf{W}}^T\mathbf{u}(0) = \mathbf{W}^T\mathbf{u}(0)$. □

Therefore, after a transition, the motor activity will converge to motor activity equal to that when the bottom-up is presented without any top-down. In other words, the influence of top-down will die out over time, and transitions to other classes are possible with an incorrect initial internal context. But there may be a "transition period" where the output action may be wrong before the recovery.

5.2.5 Irreducible Networks

In the general case, networks are irreducible, meaning there is a path over nonzero weights between any two neurons. Strictly, this means the feature-transition markov chain is irreducible. As a markov chain, $\hat{\mathbf{A}}$ is aperiodic since every state links to itself with nonzero probability; it is positive recurrent, for the same reason. Thus there exists a single stationary distribution described by the stationary probability vector π, that satisfies

$$\pi \hat{\mathbf{A}}^T = \pi \tag{5.12}$$

Since $\hat{\mathbf{A}}$ is doubly stochastic, it is easy to see that $\pi_i = 1/n$, $\forall i$. Given this result, for any irreducible $\hat{\mathbf{A}}$, the limit of $\hat{\mathbf{A}}^k$ as $k \to \infty$ approaches a uniform equilibrium distribution.

This means that, for any initial activity distribution, even if the features initially firing are all mostly associated with the same motor, the steady state excitation distribution will include all feature neurons. Additionally, excitation eventually becomes distributed evenly, no matter what the initial excitation. It seems top-down connections eventually spread activity evenly throughout the entire network.

Unfortunately, an irreducible markov chain could arise with even just one high-entropy feature — a feature neuron with nonzero connections to all motor neurons. But we cannot require strictly modular networks. It is problematic for efficiency if all features are zero entropy. It is incredibly inefficient to develop feature hierarchies for each class separately. It is probable that our visual systems can recognize so many different things through effective use of shared features, which would be associated with multiple classes.

5.2.6 Modular Networks

A network like the one in Fig. 5.4(c) can be considered modular, but not strictly in the sense we discussed earlier. True modularity is characterized as described by Bullmore and Sporns [15], where nodes within each module (or “community”) have high intra-module connectivity, but have very low connectivity to neurons in other modules. Some nodes are called "hubs", having high centrality, meaning they have short paths to many other nodes. "Provincial hubs" connect to many nodes within the same module, which "connector hubs" connect to many nodes in different modules. There are very few hubs compared to the number of non-hub nodes. A modular structure is therefore characterized by communities and a few hubs, of an intra-community or inter-community nature. Bullmore and Sporns reviewed connectivity real brain data over the brains of many species and concluded that *the archetypical brain network seems to have such a modular structure.*

In the work presented here, the motor neurons are provincial hubs. Shared features might be connecter hubs. In the network in Fig. 5.4(c), the middle feature neuron is a connector hub. In our three-layer structure, a modular network is called low-entropy network, since most feature neurons are associated with a single motor. A non-modular network then has widespread connectivity, and is called high-entropy. For this network type, $\hat{\mathbf{A}}^k$ approaches the uniform stationary distribution quickly (at small k), meaning excitation spreads nondiscriminantly rapidly. There does not seem like any good way to use top-down connections in a high-entropy network. In the next subsection, several mechanisms are proposed for control of top-down connections in a modular network.

5.2.7 Nonlinear Mechanisms for Modular Networks

Modular networks are irreducible and have a single uniform stationary distribution. But their $\hat{\mathbf{A}}^k$ approaches the stationary distribution very slowly. So, $\hat{\mathbf{A}}^k \approx \hat{\mathbf{A}}$ at small k.

We wish to prevent excitation spread that might lead to decision errors in modular networks; however we do not wish to prevent excitation spread that might fill in missing information appropriately. If each module is a group of variations of same-

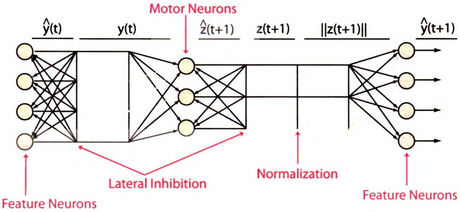


Figure 5.5: A section of an "unrolled in time" layer-two and layer-three in a network with four feature neurons and three motor neurons. Keys differences from the linear system are lateral inhibition and output normalization. Lateral inhibition, at both the feature layer and motor layer, stops the flow of low energy signals into the future and can control the spread of positive feedback. This figure shows two feature neurons inhibited at time t and one motor neuron inhibited at time $t+1$. The output normalization keeps the top-down vector on the same scale as the bottom-up. It also allows long-term memory.

type features, then the connector hub neurons can "prime" related feature types. If we see two eyes and a mouth in the right configuration, it is east to "imagine" a nose. But we do not need to imagine unrelated things.

Feature support in modular networks comes from three sources: from the sensed environment, from other neurons in its module (strong), and from connector hubs in other modules (typically weak). The influence of neurons with low support should not go to far into the future, as it could lead to errors. But we don't want to cut off neurons with moderate support, since their firing could lead to interesting and useful perception.

Several mechanisms might provide the desired effects. 1. Sigmoid activation functions. Sigmoidals are biologically supported and can suppress firing unless excitation is high enough. 2. Neuronal discharge [29]. Similar to $\beta < 0$ above, if neuron firing loses some excitation at every time step, neurons with sparse and/or temporal support will eventually stop firing. 3. Lateral inhibition. Lateral inhibition is a competitive method for cutting off weaker responses.

We have implemented a k-winners take all approximate version of lateral inhibition for experiments here. It depends on the two parameters k_1 and k_2, which are the the number of neurons that will fire on the feature layer and motor layer, respectively. Both should be greater than one, to avoid hallucination; in general k can be set to the expected amount of feature neurons that represent a class (e.g., $k_1 \doteq 15$ with $n = 400$ and $c = 26$).

Let f_{LI1} be the layer-one lateral inhibition function depending on k_1, and f_{LI2} be the layer-two lateral inhibition function depending on k_2. Now, we set $f_{LC} = f_{LCA}$, e.g., the layer computation function uses LCA:

$$\mathbf{c} = f_{LCA}(\mathbf{a}, \mathbf{b}|\mathbf{A}, \mathbf{B}) = f_{LI1}\left(\alpha\frac{\mathbf{a}}{\|\mathbf{a}\|}\mathbf{A} + \alpha\frac{\mathbf{b}}{\|\mathbf{b}\|}\mathbf{B}\right) \tag{5.13}$$

The corresponding discrete time system is:

$$\mathbf{y}(t+1) = \alpha\frac{f_{LI2}\left(f_{LI1}\left(\mathbf{y}(\mathrm{t})\hat{\mathbf{W}}\right)\right)}{\|f_{LI2}\left(f_{LI1}\left(\mathbf{y}(\mathrm{t})\hat{\mathbf{W}}\right)\right)\|}\hat{\mathbf{M}} + \alpha\frac{\mathbf{x}(t)}{\|\mathbf{x}(t)\|}\hat{\mathbf{V}}. \tag{5.14}$$

The difference between this and the linear system is twofold: 1. lateral inhibition and 2. normalization of $\mathbf{x}$ and $\mathbf{z}$. Lateral inhibition is used for reasons discussed above. Without the normalization of $\mathbf{z}$, the response $\mathbf{y}(t)$ is not controlled in its scale. Consider the use of L1-norm (so that $\mathbf{z}$ will have a probabilistic interpretation): as internally updated prior probabilities of the potential classes in the upcoming stimulus. Scale control lets the top-down influence match the bottom-up influence, which may otherwise be a problem if we have many more pixels than motors. An interesting aspect of normalization is that it allows long-term memory, by preventing decay. A byproduct of this is that there has to be explicit "null" states, as the network will not settle into zero activity. This interesting direction has not yet been explored fully.

It may be also useful to cut off weak feature responses immediately, without letting them receive any top-down boost from the motors. This strategy has somewhat of an interpretation in the laminar organization of cortical areas [94]. This is called using "paired layers", but was not done in the experiments here. Paired layers can eliminate the transition period of errors, since features with no bottom-up support will be cut off immediately.

5.3 Algorithm

This is the algorithm for running the network, without learning (frozen weights). In general development, such a thing does not occur. But we use this for experiments in this paper. We start with a mature network. Reset t to one. For $t = 1, 2, ...$

1. Sense. Set $\mathbf{x}(t)$ as before. Let $\mathbf{z}(1) = \mathbf{0}$.

2. Pre-response. Compute pre-response $\hat{\mathbf{y}}$ for all neurons on layer-one:

$$\hat{\mathbf{y}}(t) = \alpha \frac{\mathbf{x}(t)}{\|\mathbf{x}(t)\|} \mathbf{V} + \alpha \frac{\mathbf{z}(\mathbf{t})}{\|\mathbf{z}(\mathbf{t})\|} \mathbf{M}. \tag{5.15}$$

3. Lateral-Inhibition of Layer-One. Compute post-competition firing vector $\mathbf{y}(t)$ for the feature layer using approximate lateral inhibition. Again, set all neurons' firing to zero except the highest $k_1 > 1$ pre-responses. Let s_k be the k_1-th highest value of $\hat{\mathbf{y}}(t)$. Then set a neuron's response as:

$$y_i(t) = \begin{cases} \hat{y}_i(t), & \text{if } \hat{y}_i(t) \geq s_k \\ 0, & \text{otherwise} \end{cases} \tag{5.16}$$

4. Pre-response of Layer-Two. Let $\hat{\mathbf{z}}(t+1) = \mathbf{W}^T \mathbf{y}(t)$.

5. Lateral-Inhibition of Layer-Two. Here, the motor inhibition parameter k_2 will be used, which must be larger than one. Let s_k be the k_2-th highest value of $\hat{\mathbf{z}}(t+1)$. For motor neuron i, where $1 \leq i \leq m$, set its response as:

$$z_i(t+1) = \begin{cases} \hat{z}_i(t+1), & \text{if } \hat{z}_i(t+1) \geq s_k \\ 0, & \text{otherwise} \end{cases} \tag{5.17}$$

6. System Output. Take the highest firing motor to indicate the class label of the current input $\mathbf{x}(t)$.

5.4 Experiments

The networks developed in [64] and the last chapter 4, which showed topographic class grouping, have modular connectivity. Most layer-one neurons are connected to only a single motor neuron, but the border neurons between groups have significant connections to two or three motor neurons.

For these modular, top-down connections should provide a recognition rate benefit when the data is temporally continuous. We compared low-entropy networks sized 20 × 20 in two cases: top-down enabled in testing and top-down disabled in testing (i.e., $\alpha = 0$). These networks have global connectivity, i.e., each feature neuron is sensitive to all pixels.

5.4.1 Object Recognition

The MSU 25-Object dataset was used, which has views of 25 objects rotating in depth. We trained networks with 20 × 20 neurons over ten epochs using $\alpha = 0.3$ in the training phase. The images were trained in sequences, with a few empty (no object) frames in between each sequence to mimic an object being placed and taken away. The disjoint images were tested after each epoch, also presented in sequences of 40 per class with a few background frames in between each sequence. Parameters $k^{(1)} = k^{(2)} = 1$ in training, but were increased in testing. They were held constant at $k^{(1)} = 15$, $k^{(2)} = 8$ for the tests. Figure 5.6(a) shows the effect of different β in testing after 5 epochs. It can be seen that expectation leads to perfect

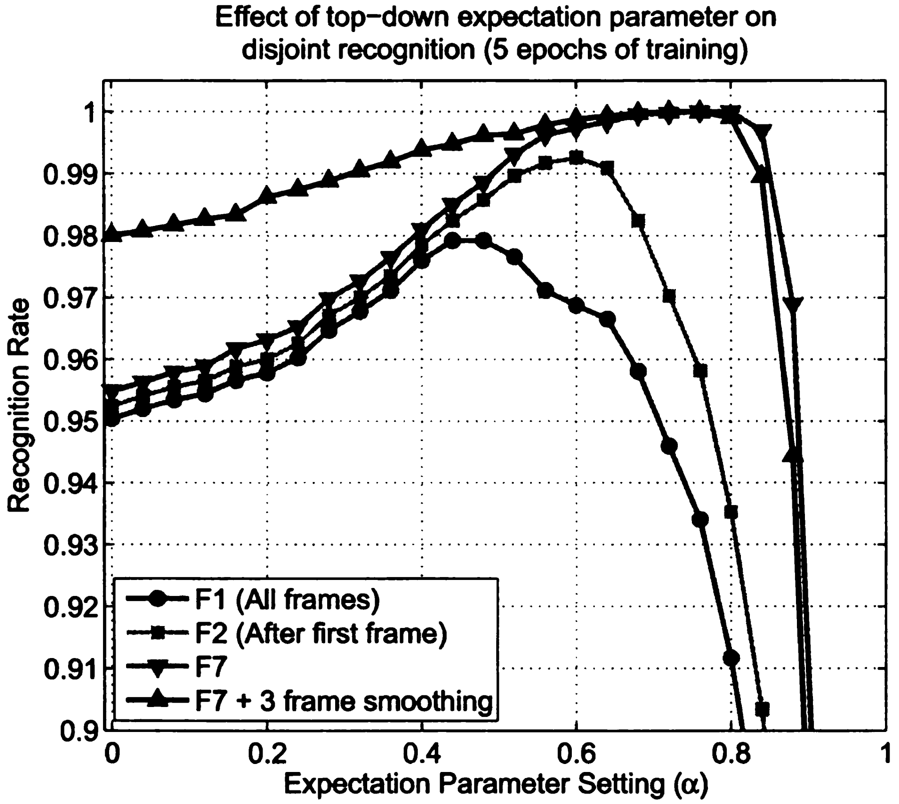


Figure 5.6: The effect of different expectation parameter values on performance. The "X" in "FX" indicates the frame to start measuring performance after a sequence shift, to a rotating object sequence. Three frame smoothing means the latest three outputs "vote" to get a single output.

performance after the transition periods. Figure 5.6(b) measures how training the same sequences over and over again can help performance. It helps a lot to see the same sequence at least twice. Figure 5.8 shows how the transition period is affected by α. Increasing expectation eventually leads to no errors except in the transition periods. But higher α will have longer transitions. It would be allowable for there to be a brief period of confusion on transitions for autonomous agents (perhaps such an effect exists in some biological agents). Such a requirement of decision after each frame seems too strict, as is we are asked "what object is that?", we will not answer without at least some delay, until we are "certain".

Since expectation takes into account the recent class history the most, the performance will be best when the probability that the next image contains the same

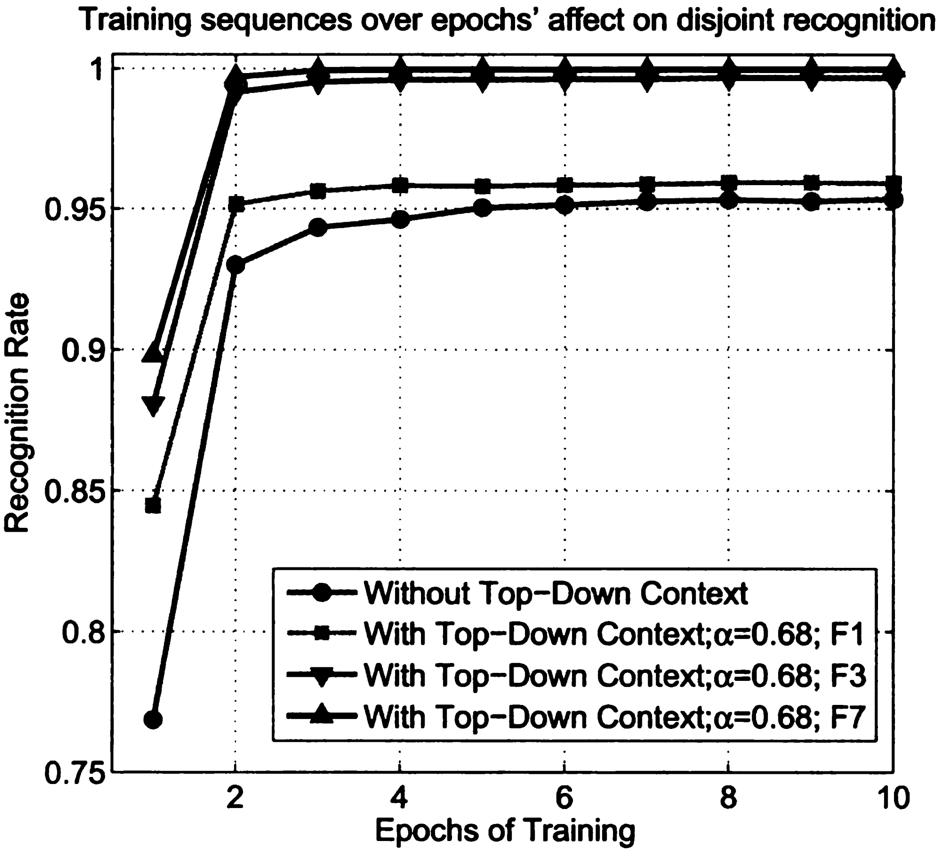


Figure 5.7: How performance improves over training epochs through all object sequences.

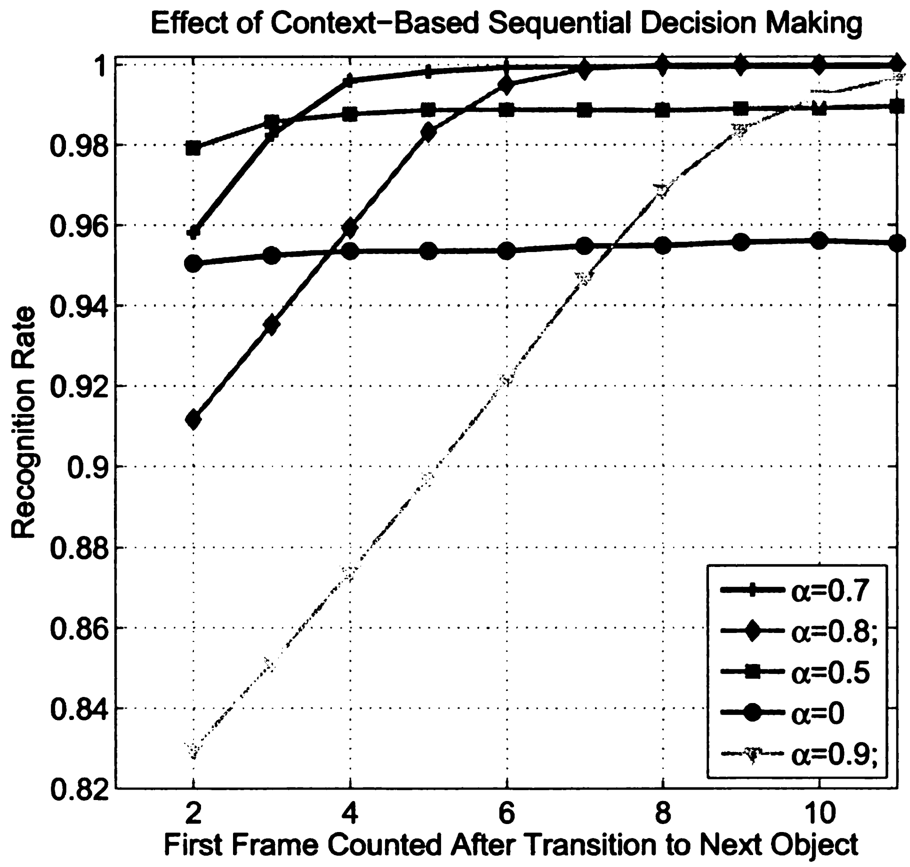


Figure 5.8: Increased expectation parameter shifts errors from bottom-up misclassifications to errors within a transition period (immediately after the class of viewed object changed).

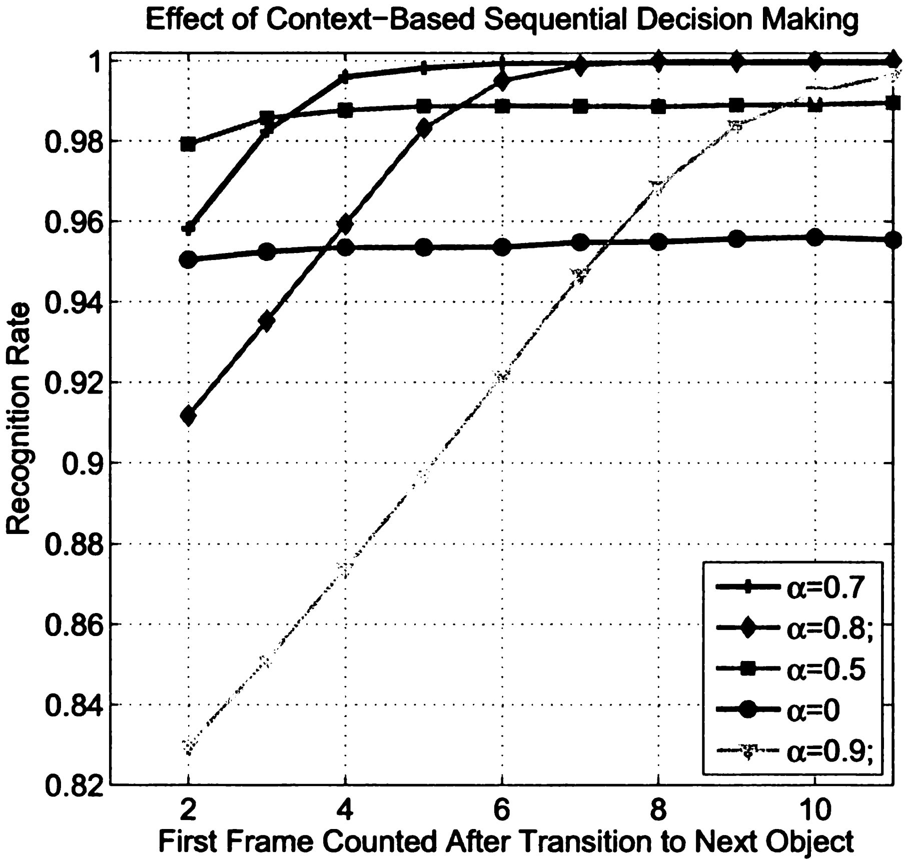


Figure 5.8: Increased expectation parameter shifts errors from bottom-up misclassifications to errors within a transition period (immediately after the class of viewed object changed).

class is high. This is called temporal continuity. Under high temporal continuity, higher expectation can be effective by reducing the effect of outlying points across the wrong decision boundary by pulling them back towards the class center, leading to more correct classifications. But it will lead to longer transition periods.

5.4.2 Vehicle Detection

Here, the above method is shown to work with local features. Additionally, global features are compared to local features on a vehicle / non-vehicle discrimination problem. Vehicles are generally decomposable into visible parts (e.g., headlight and license plate on a car). Local features should become tuned to parts, which improves the generalization power of the network, meaning the performance is better with less data. The data was collected at the GM facility using a camera mounted on the side of a test vehicle. Some examples of vehicle data used are shown in Fig. 5.9.

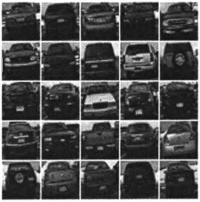


Figure 5.9: Examples of data within the vehicle class.

There were 225 vehicle images and 225 not-vehicles images, each sized 32 × 32.

To improve generalization, the network was extended to develop **local (parts-based) features** directly from the vehicle and false return data. Using the developed local features, unfamiliar objects can be recognized and classified if there are any familiar object parts (e.g., a car is recognized by a tail light) and there no need to recognize the entire object. This is critical for recognizing objects that the system has not seen before, which share some common properties with the general class they are within. It should also lead to a network that can handle occlusion.

A global network (10×10 neurons) is compared to a local network (window size of 11×11, and the local competitive area is 5×5 for each neuron), and a local network using expectation after it matures. In a global feature network, each neuron on layer-one is sensitive to every element within the sensor matrix, which has d elements (pixels). In the local network used here, the approach given in [48] is followed, where each neuron on layer-one has a $r \times r$ receptive field, where r^2 is less than d. The number of neurons is equal to the number of pixels, and each neuron's receptive field is centered on a particular pixel[2]. The competition step is also local. A neuron competes with its local region of $l \times l$ neurons. The *local* top-$k^{(1)}$ responding neurons are called winners, in the local version. The local network used here did not utilize smoothing, and β was gradually increased in training from 0 to 0.3 (in training) after 40 training samples.

The networks are initialized and trained with only 5 samples from each class. Then all samples are tested, in alternating cars and background sequences. Next, the next 5 samples are trained, and so on, until all samples are trained. This was done 10 times for each network type, using a different training order each time. In the expectation-enabled network, $\beta = 0.3$ during the testing phase, but in the other networks $\beta = 0$ in testing.

Results are summarized in Fig. 5.10. The local network does indeed do better with less data, however it eventually only does just as well as the global network. If expectation is enabled however, the performance becomes nearly perfect. Fig. 5.11 show some distinct features developed by the local network. It also shows the

[2] The image boundaries are extended with black pixels

respective spatial locations of these features.

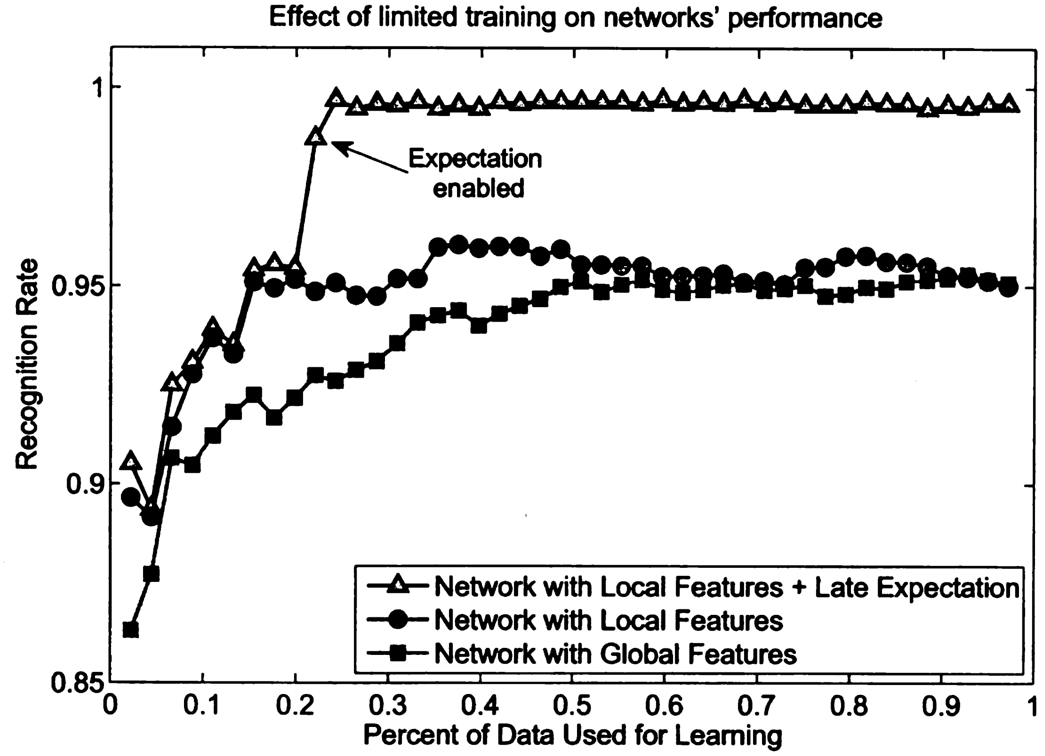


Figure 5.10: Performance of globally versus locally connected networks when data is limited. The locally connected network performs better with less data for this dataset. This may be because vehicle images can be made up of several interchangeable parts. Once training is mature, the expectation mechanism can be enabled in testing, and performance shoots up to a nearly perfect level.

5.5 Summary

This chapter describes enabling a network to keep and update an internal state **z** that biases the next activity of the feature layer. It is considered as *temporal context*: an internal "prior" that is used to bias the next decision. Ideally, this prior will bias a small subset of features highly correlated with the current state and will not bias others.

The algorithms presented are quite efficient since it does not have to learn to make decisions from several raw frames in a row. The dimensionality of such a

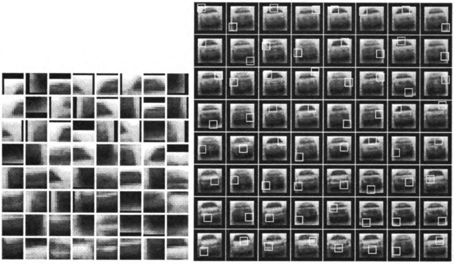


Figure 5.11: (a). Some features for the "vehicle" class, developed by the locally-connected network. This shows the *response-weighted input*, as a weighted sum of the data and the neuron responses. (b). Showing the respective locations of the features from (a). This figure shows response-weighted input, so there is nonzero sensory stimulus outside the receptive fields (the white box area), but the neurons will not respond to any stimuli outside this area.

space makes it difficult to learn such a policy. Instead it learns to make decisions from a single temporally updated parameter $\mathbf{z}$.

Since this is a positive feedback system, it is crucial to keep the spread of activity under control. For strictly modular networks, it was formally shown how positive feedback can be controlled and useful. For high-entropy networks, positive feedback potentially spreads to all neurons evenly within a short time. For modular networks — characterized by highly connected communities where different communities are connected by hubs — several nonlinear methods are discussed as ways to control positive feedback without disabling it. The network examples here control positive feedback in two key ways: 1. by developing in such a way that they are modular, 2. lateral inhibition to keep low responding neurons from spreading activity. The first point is often overlooked: a network's connectivity seems to be extremely important.

Although temporal context has been used in many task-specific models and in artificial neural networks, this is the first time where context is used as motor initiated expectation through top-down connections in a biologically-inspired general-purpose developmental network. If the motor action represents abstract meaning (e.g., recognized class) these mechanisms enable meaningful abstract expectation by such networks. We showed the effect in improving recognition rate to nearly perfect performance after a transition period when the class has first changed. Expectation's effectiveness in improving performance is crucially linked to the capacity to develop discriminating features — top-down connections may not be useful when features are not as good.

From the perspective of developmental robotics and autonomous mental development, the mechanism described in this paper can lead to more realistic learning ability for mentally developing machines. Supervised learning methods that can be applied to visual recognition of objects are formulated at a per-frame level where each training sample (frame) must be accompanied by a label. This does not take advantage of the temporal context, or the *temporal continuity* present in real-world (subject to real laws, e.g., realistic physics) video streams. When a child is taught, say to recognize numerals and letters in the classroom, there is not a continual stream of repetitive speech from the teacher speaking the names of the characters over and over. The teacher will 1. direct the child's visual attention and 2. speak the name of the character. The direction of attention hopefully ensures that the child is looking at the character continuously. For AMD, a semi-supervised learning mode can be utilized based on this chapter's method, which should significantly reduce the training load of the teacher. In that mode, expectation will be enabled and the weights not frozen. It should be useful since the common distinctions of "training phase" from "testing phase" cannot be made so easily in a real time, developmental system. Then, the teacher can sparsely provide labels.

5.6 Bibliographical Notes

Artificial neural networks can also be roughly considered as spatial and/or temporal (recurrent). A recurrent network is explicitly designed to deal with time by using feedback. A spatial network typically treats the world frame-by-frame with each frame considered independent. Spatial networks can be made to act temporally by e.g., blurring the output over time, but they are not recurrent. Recurrent networks can handle time-series prediction, where the past influences the present decision. Recurrent networks can be considered as dynamic systems.

Most networks are spatial, such as those that perform PCA [125] or ICA [8, 44]. The Self-Organizing Maps [55] are spatial, but their LISSOM extension that uses explicit lateral excitatory and inhibitory connections [73] is recurrent due to lateral excitatory and inhibitory feedback. Traditional backpropagation nets are also spatial, but backpropagation-through-time [127] converts back-prop networks into recurrent networks. Classic recurrent neural nets include Jordan nets and Elman nets [31], which utilize context units that can "remember" the last network activations. They are trained by propagating an approximation of the error gradient back through time and modifying the weights based on that. These networks suffer from long-term memory problems as the an input's influence over time either takes over network activity or decays rapidly. The long-short term memory networks [40] address this problem by ensuring an error signal propagated back through time does not blow up or vanish by using "memory cells" and "gate units". The work we present here is in the tradition of SOM and vector quantization, traditionally spatial approaches, but our approach is also a recurrent network, similarly to LISSOM. But unlike LISSOM we focus on recurrence between one layer and the next instead of on the same layer.

Douglas *et al.* described a model of neurons as electronic circuits, which distributed current proportionally using excitatory feedback. They showed that inhibitory neurons (or "neuronal discharge") was necessary for stability [29], which is similar to the conclusions here. Sejnowski [89] described how excitatory feedback could be used for prediction of temporal sequences, by using a temporally asymmetric learning rule.

Chapter 6

WWN-3: A Recurrent Network for Visual Attention and Recognition

The work presented in this article concerns a type of multilayer, multipath, recurrent, developmental network called the Where-What networks (WWN). WWNs are designed for concurrent attention and recognition, via complementary pathways leading to complementary outputs (type motor and location motor). WWNs are biologically-inspired grounded networks that learn attention and recognition from supervision. By grounded, we mean such a network is internal to an autonomous agent, which senses and acts on an external environment. WWN network models consist of two pathways for identity (what) and action-related location (where and/or how), so there are separate "motor" areas for identity and location. The motor areas connect to another later module that controls the actions of this agent. In our supervised paradigm, the agent is taught to attend by being coerced to act appropriately over many cases. For example, a teacher leads it to "say" "car" while the agent is looking at a scene containing a car, and the teacher points out the location of the car. Before learning, the agent does not understand the meaning of car, but it was coerced to act appropriately. Such action causes activation at the Type-Motor and Location-Motor in WWN-3. Top-down excitatory activity from these motor areas, which are concerned with semantic information, synchronize with bottom-up excitatory activity from the earlier areas, concerned with "physical" image information. Bidirectional co-firing is the cause of learning meaning within the network. Neurons on a particular layer learn their representations via input connections from other neurons from three locations: from earlier areas (ascending or bottom-up), from the same area (lateral), and from later areas (descending or top-down). Learning occurs in a biologically-inspired cell-centered (local) way, using an optimal local learning algorithm called Lobe Component Analysis [122].

WWN-3 utilized the following four different attention mechanisms: (1) **Bottom-up free-viewing**, (2) **Attention shift**, (3) **Top-down object-based search**, and (4) **Top-down location-based binding**. Top-down excitation is the impetus of the latter three mechanisms. In WWN-3, top-down excitation serves a modulatory role, while the bottom-up connections are directed information carriers, as are the suspected roles of these connections in the brain [14]. We'll show how an architecture feature called paired layers is important so that the top-down excitation, which we think of internal expectation, can have appropriate influence without leading to hallucination or corrupting bottom-up (physical) information with top-down semantic bias. In addition to modulation, in WWN-3, top-down connections (along with the bottom-up and lateral connections) allow a network's internal activity to synchronize. When there are multiple objects in the scene, there will be multiple internally valid solutions for type and location, but some of these solutions will actually be incorrect (mixed up). This is part of the well-known binding problem. But in WWN-3, after a bottom-up pass to select for candidate locations and candidate types, top-down location bias effectively selects a particular location to analyze and it then synchronizes with the appropriate type, similar to Treisman's idea of spotlight [109]. Top-down bias can also be introduced at the Type-Motor, causing "object search" — it picks the best candidate location given the particular type bias. After the network has settled on a particular type and location, it can disengage from the current location to try another location, or it can disengage from both location and type.

The architecture of the WWNs is divergent into two distinctive pathways for foreground identity (what) and location (where). Through development, firing of neurons further along the "What" pathway become more invariant to object position, while becoming specific to object type. The opposite is true for the "Where" pathway. Neurons earlier in each pathway represent both location and type in a mixed way.

WWN is a recurrent network; it utilizes both bottom-up and top-down connections at all areas and layers (intermediate layers included) in both the learning phase and in operation. During learning, the top-down connections are essential for the network to distinguish foreground from background in the earlier areas, and therefore are needed to learn appropriately. For the network to attend on its own, activity at one (goal-setting) or no (bias-free) motor area is imposed. In these cases, the interaction between bottom-up and top-down is essential for attention. The early area selects candidate locations and features from the input image in a bottom-up way. When one of the motor areas is imposed, information flows via top-down connections down this pathway and up the other pathway to set the other motor area. The top-down connections serve a modulatory role to pick out the biased candidates from the early area and ignore the rest. Yet the top-down connections can lead to hallucination, if there is no associated bottom-up support. To avoid this, a paired layer architecture is used, which is analyzed here in terms of signal processing theory.

6.1 Background

6.1.1 WWN Architecture

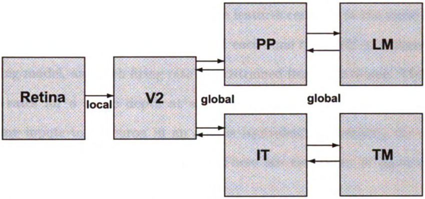


Figure 6.1: A high-level block diagram of WWN-3. The area are named after those found in our visual pathway but it is not claimed that the functions or representations are identical.

It is known that our visual system has two major pathways: ventral ("what") for object identification and dorsal ("where") that deals more with visuomotor aspects (i.e., where to reach for an object), which presumably codes an object's location. These pathways separate from early visual areas and converge at prefrontal cortex, which is known to be active in top-down attention. Prefrontal cortex connects to motor areas. WWN was built inspired by the idea of these two separating and converging pathways. Meaningful foregrounds in the scene will compete for selection in the ventral stream, and locations in the scene will compete for processing in the dorsal stream.

There are five areas of computation in WWN-3. The input image is considered as retinal activation. Instead of a multi-area feature hierarchy (ventral pathway), we use a shape-sensitive area we called V4, but we don't claim the representation is identical to V4. From this area, one path goes through the IT (inferotemporal) and TM (Type-Motor) — possibly analogous to the inferior frontal gyrus [101]. TM is concerned with object type. The other path goes through the PP (posterior parietal) area and LM (Location Motor) — possibly analogous to the frontal eye fields (FEF). LM is concerned with object location. Each of these five areas contains a 3D grid of neurons, where the first two dimensions are relative to image height and width and the third is "depth", for having multiple features centered at the same location. These neurons compute their firing rates at each time t. WWN is a discrete-time, rate-coding model, and each firing rate is constrained from zero to one. The pattern of firing rates for a single depth at any time t can be thought of as an image. Computing inputs to a neuron in an area is equivalent to sampling the image of firing rates from the input area images. There are two types of input sampling methods for an area — local or global:

- **Local input field**: V4 neurons have local input fields from the bottom-up. This means they sample the retinal image locally, depending on their position in the 2D major neural axes (ignoring depth). A neuron at location $(i,\ j)$ with receptive field size w, will take input vector from a square of sides w long, centered at location $(i + \lceil w/2 \rceil, j + \lceil w/2 \rceil)$.
- **Global input field**: Neurons with global input fields sample the entire input area as a single vector.

An architecture figure for WWN-3 is shown in Fig. 6.5. We initialized WWN-3 to use retinal images of total size 38 × 38, having foregrounds sized roughly 19 × 19 placed on them, with foreground contours based on the object's contours. V4 had 20 × 20 × 3 neurons, with bottom-up local input fields (of 19 × 19) at different locations on the retina (based on the neurons' 2D locations), and top-down global receptive fields. PP and IT also had 20 × 20 neurons and had bottom-up and top-down input fields that were global. LM had 20 × 20 neurons with global bottom-up input fields, and TM had 5 × 1 neurons (since there were 5 classes) with global bottom-up receptive fields.

6.1.2 Attention Selection Mechanisms at a High-Level

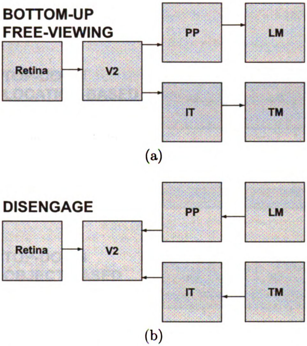
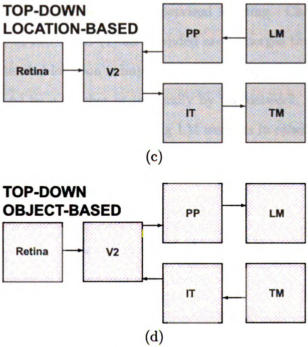


Figure 6.2: WWN accomplishes different modes of attention by changing the directions of information flow. (a) For bottom-up stimulus driven attention, information flows forward from pixels to motors. (b) In order to disengage from the currently attended object, an internal suppression is applied at the motor end, and a diffuse top-down excitation will cause a new foreground to be attended to in the next bottom-up pass.

Selective attention is not a single process; instead, it has several components. These mechanisms can be broken down into orienting, filtering and searching. These are not completely independent, but the distinctions are convenient for the following discussion. The Where-What network makes predictions about how each of the mechanisms could work, and, in the following, we will discuss how these mechanisms work in WWN-3.



(c)

(d)

Figure 6.3: WWN accomplishes different modes of attention by changing the directions of information flow. This continues the last figure. (c) Top-down location-based attention (orientation) occurs due to imposed location motor and information flows back to V4 and forward to the type motor. (d) Top-down type-based attention (object search) occurs due to imposed type motor and information flows back to V4 and forward to the location motor.

Orienting

Orienting is the placement of attention's location. In covert orienting, one places one's focus at a particular location in the visual field without moving the eyes; this is different from overt orienting, which would require the eyes to move. Covert orientation is realized in WWN based on sparse firing (e.g., winner-take-all or WTA) in the LM area. LM neurons' firing is correlated with different attention locations on the retina, which emerged through supervised learning. Changes in attended location can occur in two ways: (1) an attended area emerges through feedforward activity, (2) an attended location is imposed ("location-based" top-down attention, which could be done by a teacher or internally by the network, or (3) a currently attended location is suppressed, by boosting LM neurons in other areas. The effect of this is to de-engage attention, and shift to some other foreground.

Filtering

Attention was classically discussed in terms of filtering [13, 67, 106]. In order to focus on a certain item in the environment, the "strength" of the other information seems to be diminished. In WWN-3, there are multiple passes of filtering. 1. Early filtering. This is done without any top-down control. As WWN-3 is developmental, the result of early filtering depends totally on the filters that were developed in the supervised learning phase. Then, responses of these developed V4 neurons in WWN are an indicator of what interesting foregrounds there are in the scene. If there is no top-down control, internal dynamics will cause to converge to a steady state with a single neuron active in each of TM and LM, representing the type (class) and location of the foreground. A single feedforward pass is thought to be enough in many recognition tasks. But if there are multiple objects, attention seems to focus on each individually [109].

Another filtering process, based on top-down excitation, occurs on the result of the first-pass filtering. 2. Biased filtering. This "second-pass" filtering is in the service of some goal, such as searching for a particular type of foreground. The first-pass filtering has coded the visual scene into firing patterns representing potentially meaningful foregrounds. Binding after the first pass would be done in a purely feedforward fashion, and an incorrect result could then emerge at the motor layers. The second-pass filtering is due to top-down expectation. It re-codes the result of first pass filtering into biased firing patterns, which then influence the motor areas. The second-pass allows the network to synchronize its state among multiple areas. For example, feedback activity from the Location-Motor causes attention at a particular location, which causes the appropriate type, at that location, to emerge at the Type-Motor.

Searching

Searching for a foreground type is realized in WWN-3 based on competitive firing (e.g., WTA) of the Type-Motor (TM) neurons. Similar to the link between retinal locations and LM neurons, correlations between foreground type and TM neurons are established in the training phase. Along the ventral pathway, the location information is gradually discarded. Top-down type-based attention will cause type-specific activation to feed back from TM to V4, biasing first-stage filtered foregrounds that match the type being searched for. Afterwards, the new V4 activation feeds forward along the Where pathway, and will cause a single location to become attended based on firing in the location motor area.

6.1.3 Binding Problem

When multiple objects are present in the scene, the binding problem is an issue for WWN. Since location and type are dealt with separately, there are multiple internally valid solutions that are incorrect. Through top-down connections, we can

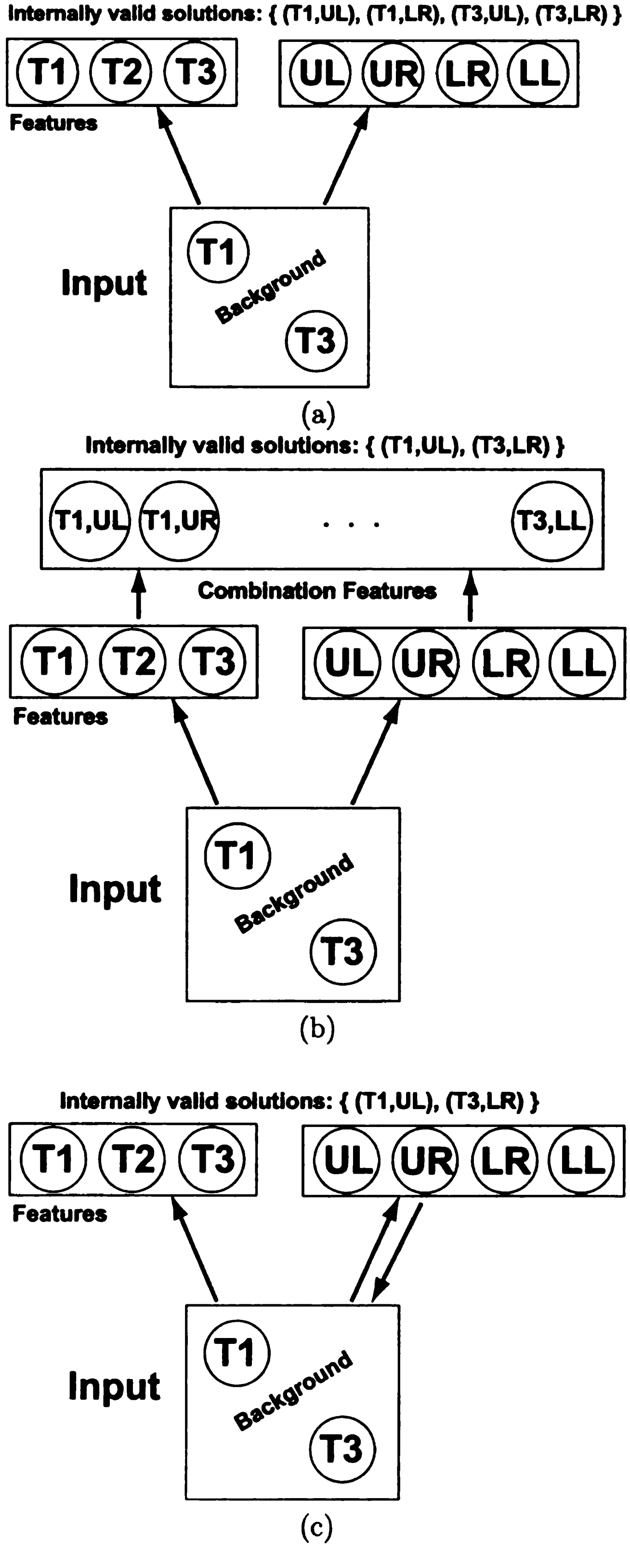


Figure 6.4: Toy example illustrating the "coarse" binding problem.

handle the "coarse" binding problem through synchronization. See Fig 6.4. In (a), separable features avoid a combinatorial explosion of representation but introduce the problem of how to reintegrate the information. The two types of information here are location and type. When there are two types in two different locations in the input, a feedforward network using winner-take-all at the two output layers can give four possible internally valid solutions. But two of these are wrong. (b) Possible solution: combination neurons. Another winner-take-all layer of units that represent each combination of features can solve this problem, but this leads to the very combinatorial explosion we were trying to avoid by using separable feature layers. Additionally, every single combination will have to be individually learned. (c) A solution avoiding the combinatorial explosion: synchronization using top-down connections. The initial feedforward activity is an initial guess of location and type, and this allowed to be inconsistent. Top-down connections from the location area bias a particular area on the image so that a incorrect type is suppressed and the correct type chosen. This is a basic way to implement a "spotlight" of attention. There still may be binding required within the spotlight, if there are occlusions or transparency. Top-down connections within a visual hierarchy might solve the full binding problem (e.g., form biases lower-layer edges).

In WWN's bias-free mode, the binding problem has to be dealt with. However, using combination neurons are problematic since they introduce the combinatorial explosion problem [107]; additionally they do not allow generalization properly [116]. In WWN, the coarse binding problem (not consideration occlusions or transparency) is handled by having a feedforward pass, then letting information flow from the location motor back towards the early feature layer and up the other pathway to the type motor. In this way, internal consistency is verified.

6.2 Concepts and Theory

Here, we discuss how bottom-up and top-down information are integrated in the Hebbian network (see Fig. 6.5).

Paired Input

For some neuron on some layer, denote its bottom-up excitatory vector by $\mathbf{x}$, and its top-down excitatory vector as $\mathbf{z}$. We used paired input:

$$\mathbf{p} \leftarrow \left(\rho\frac{\mathbf{x}}{\|\mathbf{x}\|}, (1-\rho)\frac{\mathbf{z}}{\|\mathbf{z}\|}\right) \tag{6.1}$$

where $0 \leq \rho \leq 1$. The parameter ρ allows the network to control the influence of bottom-up vs. top-down activation, since the vector normalization fundamentally places bottom-up and top-down on equal ground. Setting $\rho = 0.5$ gives the bottom-up and top-down equal influence.

Paired Layers

In Fig. 6.5, we show three internal components within each major area of V4, IT, and PP. This internal organization is called paired layers. Paired layers handle and store bottom-up information separately from the top-down boosted bottom-up information. The paired layer organization is inspired by the six-layer laminar organization which is found in all cortical areas [18], but it has been simplified to its current form. Further discussion of this architecture feature is found in [94].

Paired layers allow a network to retain a copy of its unbiased responses internally. Without paired layers, top-down modulatory effects can corrupt the bottom-up information. Such "corruption" might be useful, when the internal expectation (we consider top-down excitation as an internal expectation) relates to something in the visual scene. However, there are times when such expectations are violated. In such

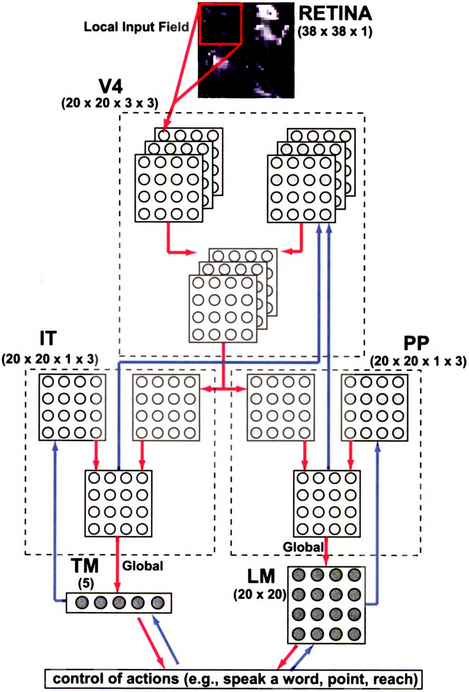


Figure 6.5: WWN-3 system architecture. The V4 area has three layers (depths) of feature detectors at all image locations. Paired layers: each area has a bottom-up component, a top-down component, and a paired (integration) component. Within each component, lateral inhibitory competition occurs. Note that in this figure top-down connections point up and bottom-up connections point down.

instances, the internal expectation is in opposition to reality. Then, the incorrect feedback can potentially lead to false alarms, or hallucinations.

To explain further, we present the following formalism to explain how paired layers are implemented. Given a neuronal layer l, let $\mathbf{V}$ be the matrix containing bottom-up column weight vectors to neurons in this layer, and $\mathbf{M}$ be the matrix containing top-down column weight vectors to the layer, where each of the column vectors in these matrices are normalized. $\mathbf{X}$ is a bottom-up input matrix: column i contains the input activations for neuron i's bottom-up input lines. $\mathbf{Z}$ is the top-down input matrix. For the following, $\mathbf{X}$ and $\mathbf{Z}$ are also column normalized. We use $\mathbf{diag(A)}$ to mean the vector consisting of the diagonal elements of matrix $\mathbf{A}$.

Non-paired layers:. First compute layer l's pre competitive response $\hat{\mathbf{y}}$:

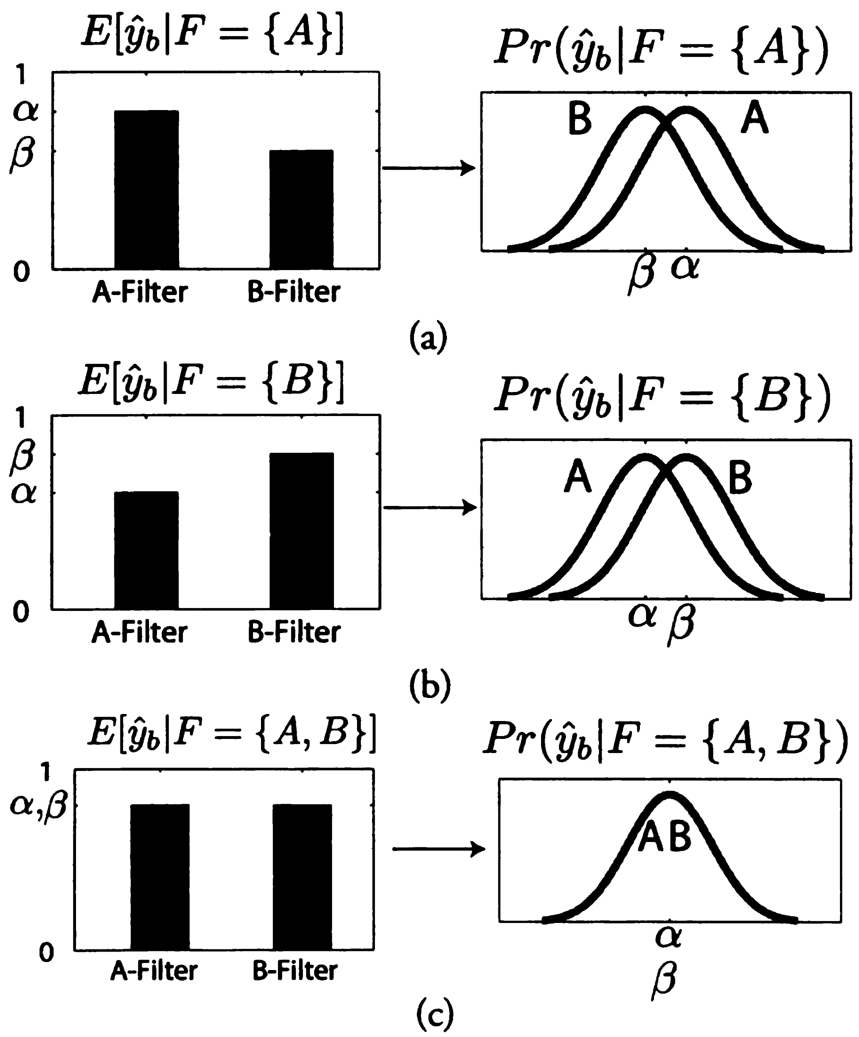
$$\hat{\mathbf{y}} = \rho \; \mathbf{diag}(\mathbf{V}^T\mathbf{X}) + (1-\rho) \; \mathbf{diag}(\mathbf{M}^T\mathbf{Z}) \tag{6.2}$$

where ρ and $1 - \rho$ are positive weights that control relative bottom-up and top-down influence. The post-competitive firing rate vector $\mathbf{y}$ of layer l is computed after lateral inhibition function f, controlled by the layer's sparsity parameter k. The top k firing neurons will fire and others have firing rate set to zero, giving a sparse response:

$$\mathbf{y} = g\left(f\left(\hat{\mathbf{y}},\ k\right)\right) \tag{6.3}$$

The lateral inhibition and sparse coding method f is achieved by sorting the components of the pre-response vector $\hat{\mathbf{y}}$. Let s_1 be the highest value, s_k be the k-th highest. Then set a neuron's response as:

$$y_i \leftarrow \begin{cases} \hat{y}_i, & \text{if } \hat{y}_i \geq s_k \\ 0, & \text{otherwise} \end{cases} \tag{6.4}$$



(a)

(b)

(c)

Figure 6.6: Interaction between bottom-up and top-down in a toy attention problem. There are two foregrounds A or B. The foreground set is F. This system has A-detectors and B-detectors. Consider the figures on the left. The expectation of pre-response of an A-filters is shown to the left and expectation for B-filters on the right. Assume the corresponding pre-response distributions are Gaussian, which are shown on the right side. A key assumption is that the pre-response alone does not lead to the best detection, shown as overlap in the distributions. When both foregrounds are visible, the system is not biased to detect one or the other, unless a top-down goal is utilized (c).

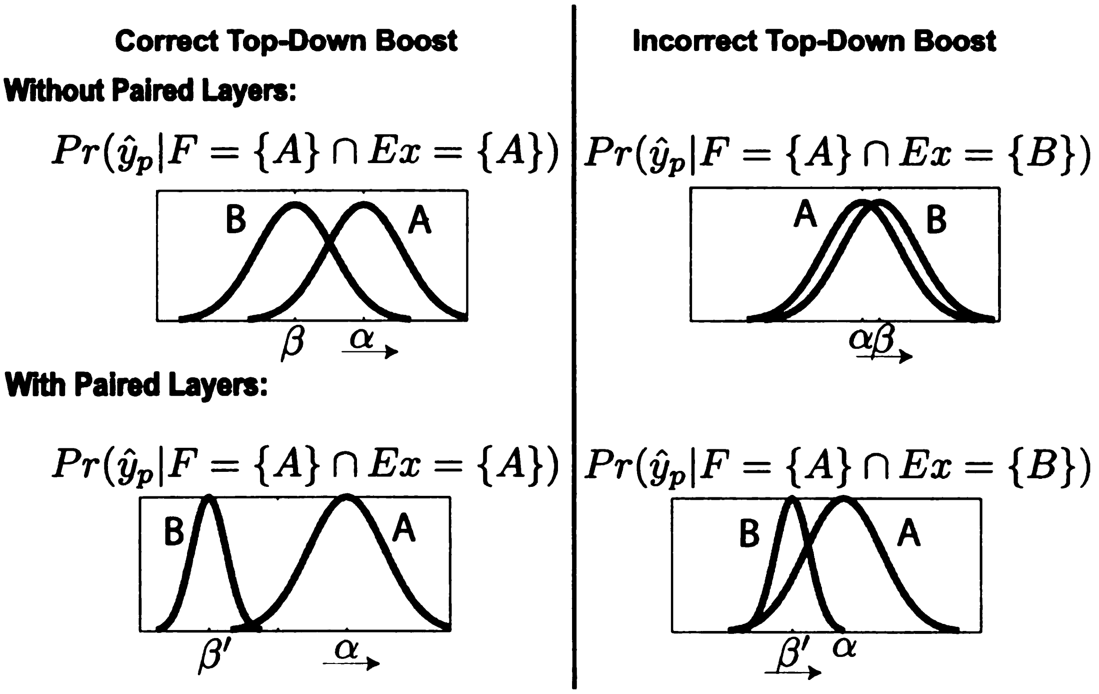


Figure 6.7: Benefit of paired layers in attention selection. The pre-response distributions of A-filters vs. B-filters are shown, when the A foreground is the only one visible, but when top-down expectation Ex either biases A (correct) or B (incorrect). Top-down boosting in cooperation with the true state (on the left) leads to higher discriminability by moving the two means farther away, which is very useful. But if the expectation is wrong (on the right), it can be drastically more difficult to detect the true state. If paired layers are used, lateral inhibition is applied first, so that many of the B-detectors will have zero response before the top-down boost. Then, an incorrect boost can be managed.

where g is a threshold function that prevents low responses after competition.

Paired layers: With paired layers, the firing rate vector $\mathbf{y}$ of layer l neurons are computed after lateral inhibition of both the bottom-up part $\hat{\mathbf{y}}_b$ and top-down part $\hat{\mathbf{y}}_t$ separately, and additional lateral inhibition for the integrated response:

$$\begin{aligned} \mathbf{y}_b &= g\left(f\left(\mathbf{diag}(\mathbf{V}^T\mathbf{X}),\ k_b\right)\right) \\ \mathbf{y}_t &= g\left(f\left(\mathbf{diag}(\mathbf{M}^T\mathbf{Z}),\ k_t\right)\right) \\ \mathbf{y} &= g\left(f\left(\rho\,\mathbf{y}_b + (1-\rho)\,\mathbf{y}_\mathbf{t},\ k_p\right)\right) \end{aligned} \tag{6.5}$$

A key idea is that the lateral inhibition causes sparse firing within the bottom-up layer and sparser firing in the paired layer (e.g., $k_b = 20$ and $k_p = 10$ where there are 1200 neurons). The top-down layer is generally not as sparse (e.g,. $k_t = 200$) so that it might reach as many potentially biased neurons as possible. In a non-paired layer, such diffuse top-down biasing can match up with many relatively weak responding filters (if there was no top-down influence) and boost them above the stronger filters that have more bottom-up support. But the intermediate competition step in the paired layer method ensures the diffuse top-down biasing will not significantly boost the relatively weak filters since they were already eliminated in bottom-up competition. Filters with support from both bottom-up and top-down thus receive the most benefit.

Both bottom-up and top-down are highly local in both space and time — they are highly spatiotemporally sensitive. The input image (bottom-up) could change quickly, or the "intent" of the network (top-down) could change quickly. In either case, a paired layer adapts quickly.

Take the following simplified case for understanding. Denote the foreground set of a visual scene by F. There are two types of interesting foregrounds — call these foregrounds A and B. For detection and attention, our network has a set of filters — two at each possible location; one to detect foreground A, and one to detect foreground B.

We must assume the expected pre-competitive response of an A-filter is higher than a B-filter when A is actually present, and B is not (and vice versa). Assume $E[\hat{y}_{b,i}] = \alpha$ when filter i is an A-filter, and $E[\hat{y}_{b,i}] = \beta$ if filter i is a B-filter. Thus, $\alpha > \beta$ if $F = \{A\}$, $\alpha < \beta$ if $F = \{B\}$. In this example, let $\alpha = \beta$ when $F = \{A, B\}$. This is graphically shown in Fig. 6.6.

Assume $Pr(\hat{y}_b)$ is Gaussian. The mean will be α for A-detectors and β for B-detectors. See Fig 6.7. If they have equal standard deviations σ, then the discriminability is defined as

$$d' = \frac{|\alpha - \beta|}{\sigma} \tag{6.6}$$

For success of detection, it is desirable to maximize d'.

The effect of top-down feedback is one of two possibilities. If the system expects foreground A, A-filters are selectively boosted and α is increased. One can see how this will increase d' when $F = \{A\}$ or $F = \{A, B\}$ but decrease d' when $F = \{B\}$. Otherwise, if the system expects foreground B, thus B-filters are selectively boosted and β is increased.

Looking at the figures in Fig. 6.7, observe that when top-down expectation is applied in accordance with the true state, $|\alpha - \beta|$ increases, and thus so does d'. If $F = \{A, B\}$, top-down feedback again increases d', presumably beneficially. However, when top-down feedback is applied in opposition to the true state, d' will decrease. This is natural, but we wish to minimize this decrease. In other words, to make the probability of false alarm $Pr(FA)$ as low as possible. Fig. 6.7(b) contrasts using paired layers to without, and shows why we expect $Pr(FA)$ will be less when paired layers are used. A boosted weak response leads to a higher false alarm rate, but if we cut off weak responses before they are boosted, many incorrect filters will have zero response, leading to a lower mean Gaussian with lower standard deviation (since many elements are zero). Then, the incorrect boosting is not as harmful, as false alarms will be lessened.

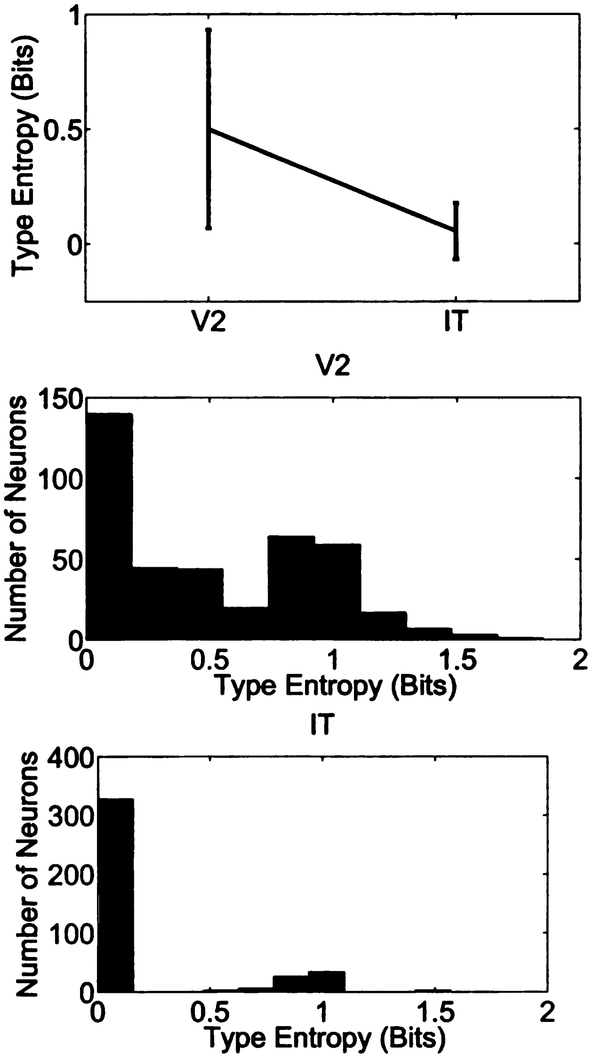


Figure 6.8: Measured entropy along the "What" WWN pathways. In the what pathway, there is a clear entropy reduction from V2 to IT. The top-down connections enabled this emergence of discriminating representation to occur.

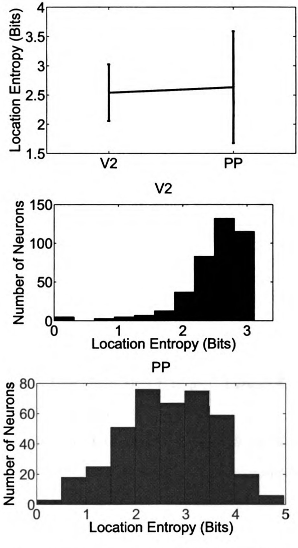


Figure 6.9: Measured entropy along both the "Where" WWN pathway. However, there is not much of an entropy reduction along the where pathway, hovering around 2.7 bits, which is about 6.5 different pixels of inaccuracy — a little smaller than a 3 × 3 pixel neighborhood. We guess that there is an accuracy ceiling for our current method that we ran into using 400 where classes (compared to 5 what classes) with the 400 neurons in PP and 1200 neurons in V2.

6.2.1 Learning Attention

Each bottom-up weight for a V4 neuron was initialized to a randomly selected 19×19 training foreground, as seen in Fig 6.15(b). This greatly quickens training by placing the weights close to the locations in weight space we expect them to converge to. Initial bottom-up weights for IT, PP, TM, LM were randomly set. Top-down weights to V4, IT and PP were set to ones (to avoid initial top-down bias)[1].

WWN learns through supervised learning externally and local learning internally. The Type-Motor and Location-Motor areas were firing-imposed per-sample in order to train the network. There are c neurons in TM, one for each class, and we used 20×20 neurons in LM, one for each attention location. Each areas' neurons self-organize in a way similar to self-organizing maps (SOM) [55], but with combined bottom-up and top-down input $\mathbf{p}$, and using the LCA algorithm for optimal weight updating. WWN's top-down connections have useful roles in network development (training). They lead to discriminant features and a motor-biased organization of lower layers. Explaining these developmental effects are out of the scope of this paper, but have been written about elsewhere [64]. Further focus on learning and development in WWN is presented in [47].

For a single area to learn, it requires bottom-up input $\mathbf{X}$, top-down input $\mathbf{Z}$, bottom-up and top-down weights $\mathbf{V}$ and $\mathbf{M}$, and the parameters ρ (controlling influence of bottom-up versus top-down), k_b (the number of neurons to fire and update after competition in the bottom-up layer), k_t (the same for the top-down layer), and k_p (for the paired layer). This area will output neuronal firing rates $\mathbf{y}$. It updates neuronal weights $\mathbf{V}$ and $\mathbf{M}$.

The non-inhibited neurons update their weights using the Hebbian-learning LCA updating [122]:

[1] Setting to initial positive values (nonzero) mimics the initial overgrowth of connections in early brain areas, later pruned by development.

$$\mathbf{v}_i \leftarrow \omega(\eta_i)\ \mathbf{v}_i + (1-\omega(\eta_i))\ \mathbf{x}_i\ y_i \tag{6.7}$$

where the plasticity parameters $\omega(\eta_i)$ and $(1-\omega(\eta_i))$ are determined automatically and optimally based on the neuron's updating age η_i. A neuron increments its age when it wins in competition. Learning rate of each neuron is a function of its firing age[2]. This learning is Hebbian as the strength of updating depends on both presynaptic potentials (e.g., $\mathbf{x}_i$) and postsynaptic potentials (e.g., y_i).

Denote the per-area learning algorithm we have discussed as *LCA*. To train the whole WWN, the following algorithm ran over three iterations per sample. Let $\theta = (k_b, k_t, k_p, \rho)$.

1. $(\mathbf{y}^{V4}, \mathbf{V}^{V4}, \mathbf{M}^{V4}) \leftarrow LCA(\mathbf{X}^{V4}, \mathbf{Z}^{V4}, \mathbf{V}^{V4}, \mathbf{M}^{V4}, \theta^{V4})$
2. $(\mathbf{y}^{IT}, \mathbf{V}^{IT}, \mathbf{M}^{IT}) \leftarrow LCA(\mathbf{X}^{IT}, \mathbf{Z}^{IT}, \mathbf{V}^{IT}, \mathbf{M}^{IT}, \theta^{IT})$
3. $(\mathbf{y}^{P}, \mathbf{V}^{P}, \mathbf{M}^{P}) \leftarrow LCA(\mathbf{X}^{P}, \mathbf{Z}^{P}, \mathbf{V}^{P}, \mathbf{M}^{P}, \theta^{P})$
4. $(\mathbf{y}^{TM}, \mathbf{V}^{TM}, \mathbf{0}) \leftarrow LCA(\mathbf{X}^{TM}, \mathbf{0}, \mathbf{V}^{TM}, \mathbf{0}, \theta^{TM})$
5. $(\mathbf{y}^{LM}, \mathbf{V}^{LM}, \mathbf{0}) \leftarrow LCA(\mathbf{X}^{LM}, \mathbf{0}, \mathbf{V}^{LM}, \mathbf{0}, \theta^{LM})$

A few more items to note on training: (1) Each area's output firing rates $\mathbf{y}$ is sampled to become the next area's bottom-up input $\mathbf{X}$, and the previous area's top-down input $\mathbf{Z}$. (2) For V4, each top-down source from the What or Where path is weighted equally in setting $\mathbf{Z}^{V4}$. (3) We used a supervised training mechanism ("pulvinar"-based training [47]) to bias V4 to learn foreground patterns: we set its $\mathbf{Z}$ based on the firing of the LM area — only neurons with receptive fields on the foreground would receive a top-down boost. This is not quite a "skull-closed" training method, but it is used since we have a limited size network. (4) For PP (denoted as "P" above) and IT areas, we used 3 × 3 neighborhood updating in the vein of self-organizing maps in order to spread representation throughout the

[2] The above equation is for bottom-up weights. For top-down weights, substitute in $\mathbf{z}_i$ for $\mathbf{x}_i$ and $\mathbf{m}_i$ for $\mathbf{v}_i$.

layer. This was done for the first two epochs. (5) The above algorithm is a forward-biased algorithm, in the sense that it takes less time for information to travel from sensors to motors (one iteration) than from motors to V4 (two iterations). Therefore, weight-updating in V4 only occurred on iterations two and three for each image. (6) Parameters: ρ^{V4} was set to 0.75 (bottom-up activity contributed 75% of the input), and $\rho^{IT} = \rho^{P} = 0.25$. In training, all values of k were set to one. This ensured a sparse representation to develop.

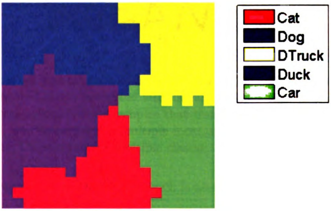


Figure 6.10: Class representation that developed in the IT area. Observe there are five different areas; one per class. Neurons along the border will represent two (or more) classes. This is illustrated in the entropy histograms above by the small group of neurons with about 1 bit of entropy (two choices).

6.2.2 Entropy Reduction

Neurons further along the What pathway become more invariant to object position, while becoming specific to object type. The opposite is true for the Where pathway. Neurons earlier in each pathway represent both location and type in a mixed way. For more information on learning, see [47]. Tests in feedforward mode with a single foreground in the scene showed the recognition and orientation performance

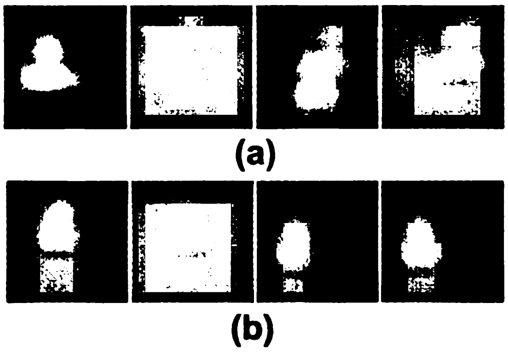


Figure 6.11: Internal representation in the IT and PP areas in the developed network. (a) Response-weighted input (weighted sum of samples) for four neurons from IT. This shows the weighted average of the samples that these neurons fired for. These represent the "duck" class, and some fire for multiple locations. (b) Response-weighted input for some PP neurons. These represent multiple classes, but a single location.

improved after epochs of learning (an epoch is an entire round of training all possible samples).

The specificity of a layer can be measured by its firing entropy relative to a set of classes which are either types or locations. As an example, consider that we measure the response of a single neuron in a "what" layer over a set of stimuli from different classes. If this neuron only fired when one of the classes was present in the stimulus and not for the others, it is considered type-specific. If this neuron only fired for one of the classes, and additionally that class could be placed in multiple locations, that neuron is also considered location-invariant. We measure entropy for a neuron by the following: $-\sum_i^c Pr(i)\log_2(Pr(i))$, where $Pr(i)$ is the probability neuron fired for class i (there are c classes). We can then characterize the entropy of the entire layer or area by measuring it for each neuron and taking the average.

The base c logarithm ensures that a neuron's maximum entropy is one (firing equally for each class). Probability is measured approximately. We tested the net-

work over a set of stimuli, each containing one of the c classes, and this neuron has fired for n of them. The probability it fires for class i ($p(i)$) is measured as the number of stimuli containing class i this neuron fired for divided by n.

After learning, we measured the entropy along both pathways respective to their particular information types. Entropy reduction for type was very apparent in the what pathway (see Fig. 6.8), but entropy reduction for location was not seen in the where pathway. This is probably due to the sheer number of where classes (400) compared to the what classes (5) and due to the high location specificity of the V4 neurons (due to the "pulvinar" supervision method used). Fig. 6.10 shows how IT organized a class-grouped representation (due to top-down connections and 3×3 updating), allowing near-zero entropy to occur in IT. Here, only the border neurons fire for more than one class, reflected in Fig. 6.8. Fig. 6.11 shows some internal representation for a few IT and PP neurons.

6.2.3 Attention Selection Mechanisms

Through training, WWN-3 becomes sparsely and selectively wired for attention. Afterwards, manipulation of parameters allows information to flow in different ways. The changing of information flow direction is a key to its ability to perform different attention tasks. Specifically, it involves manipulating the ρ parameters (bottom-up vs. top-down within an area) and γ (percentage of top-down to V4 from IT vs. PP).

In WWN-3, we examined four different attention modes. Free-viewing mode is completely feedforward. It quickly generates a set of candidate hypotheses about the image based on its learned filters. But there can be no internal verification that the type and location that emerge at the motors match (binding problem). Free-viewing mode is necessary to reduce complexity of an under-constrained search problem, but it cannot solve the problem itself. Another mode, top-down location-based binding acts as a spotlight. It constrains firing at LM to a winner location neuron and selects

for TM appropriately through top-down bias from LM to V4 and bottom-up from V4 to TM. The top-down object-based attention mode allows the network to act in search mode. It constrains firing at TM to a winner neuron and selections for LM appropriately through top-down bias from TM to V4 and bottom-up from V4 to LM. The forth mode involves a disengage from the currently attended location or both currently attended location and type.

Multiple objects introduce the binding problem for WWN-3. Another problem is the hallucination problem, which could occur when the image changes (containing different foregrounds). The following rules for attention allow WWN-3 to deal with multiple objects and image changes. Whenever a motor area's top neuron changes suddenly, switch to the corresponding top-down mode if it is a strong response, or switch to free-viewing mode if it is a weak response. Whenever both motor areas' top neurons change suddenly to strong responses, go into the top-down location-based mode. If the network focuses on a single location and type for too long, disengage from the current location. The following parameter settings specify how this was done in WWN-3. In all modes we set $k_b^{V4} = 8$ and $k_p^{V4} = 4$.

1). **Bottom-up free-viewing**: In forward mode, there is no top-down since the network does not yet have any useful internal information to constrain its search. WWN-3 sets ($\rho^{V4} = \rho^{P} = \rho^{IT} = 1$).

2). **Top-down searching (object-based)**: In this mode, information must travel from the TM down to V4 and back up to the LM. If $\gamma = 1$, all top-down influence to V4 is from the what path. Set $\gamma = 1$, $\rho^{IT} = 1$, $\rho^{P} = 0$, and $\rho^{V4} = 0.5$. k_t^{V4} is large (50% of neurons) to allow wide-spread top-down bias. For IT and PP, k_b is set small (up to 10% of neurons), for sparse coding, while k_t and k_p must be large enough to contain all neurons that may carry a bias. For example $k_t^{IT} = n/c$ where $c = 5$ (classes) and $n = 400$ neurons if we assume an equal number of neurons per class.

3). **Top-down location-based**: In this mode, information must travel from LM to V4 and back up to TM. Thus, the network sets $\gamma = 0$, $\rho^{IT} = 0$, $\rho^{P} = 1$, and $\rho^{V4} = 0.5$. The k parameters are set the same as for object-based attention.

4). **Location and type attention shift**: In this mode, the network must disengage from its current attended foreground to try to attend to another foreground. To do so, the current motor neurons that are firing are inhibited (for the location motor, an inhibition mask of 15×15 width was used) while all other neurons are slightly excited until the information can reach V4 (two iterations). The information flows top-down from motors to V4 ($\rho^{IT} = \rho^{PP} = 1$). The top-down activation parameters k_t must be set much larger since all neurons *except* the current class should be boosted. After two iterations, it re-enters free-viewing mode.

Is it plausible to have multiple different attention modes? Computationally, attention and recognition is a "chicken-egg" problem, since, for attention, it seems recognition must be done, and for recognition, it seems attention (segmentation) must be done. The brain might deal with this problem by using complementary pathways and use of different internal modes to enforce internal validity and synchronization. For example, Treisman famously showed [109] that there is a initial parallel search followed by a serial search (spotlight), which binds features into object representations at each location. It seems that after feedforward activity, a top-down location bias emerges to focus on a certain spot.

6.3 Experiments

Each input sample to WWN-3 contains one or more foregrounds superimposed over a natural background. The background patches were 38×38 in size, and selected from 13 natural images[3]. The foregrounds were selected from the MSU 25-Objects

[3] Available from http://www.cis.hut.fi/projects/ica/imageica/

dataset [64] of objects rotating in depth. The foregrounds were normalized to 19 × 19 size square images, but were placed in the background so that the gray square contour was eliminated (masked). Three training views and two testing views were selected per each of the five classes. The classes and within-class variations of the foregrounds can be seen in Fig. 6.15(b). Five input samples, with a single foreground placed over different backgrounds, can be seen in Fig. 6.15(a).

6.3.1 WWN-3 Learns

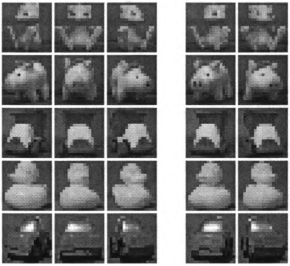


Figure 6.12: The foregrounds used in the experiment. There are three training (left) and two testing (right) foregrounds from each of the five classes of toys: "cat", "pig", "dump-truck", "duck", and "car".

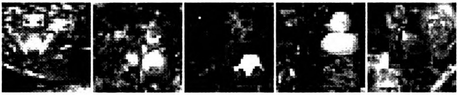


Figure 6.13: Sample image inputs.

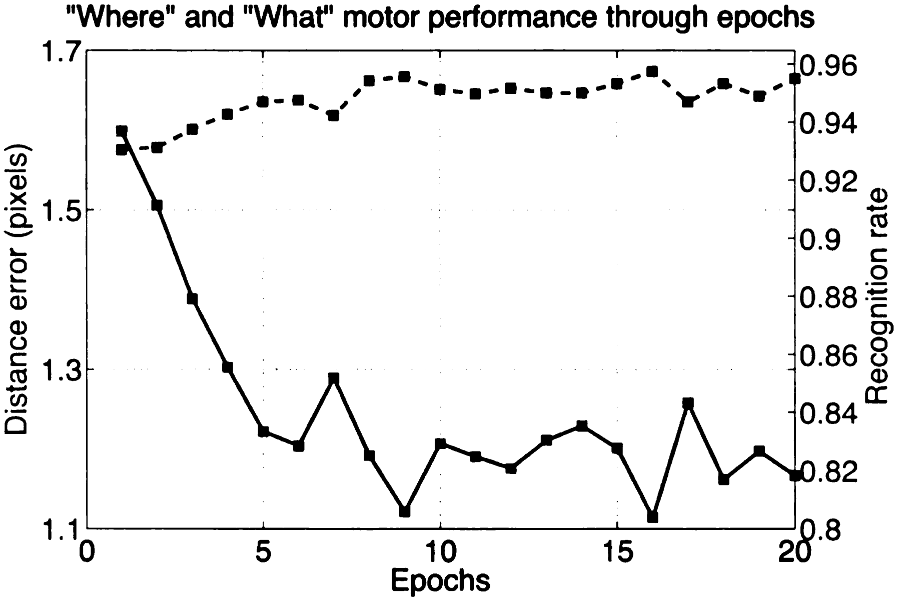


Figure 6.14: Performance results on disjoint testing data over epochs.

The training set consisted of composite foreground and background images, with one foreground per image. We trained every possible training foreground at every possible location (pixel-specific), for each epoch, and we trained over many epochs. So, there are 5 classes × 3 training instances × 20 × 20 locations = 6000 different training foreground configurations. After every epoch, we tested every possible testing foreground at every possible location. There are 5 × 2 × 20 × 20 = 4000 different testing foreground configurations.

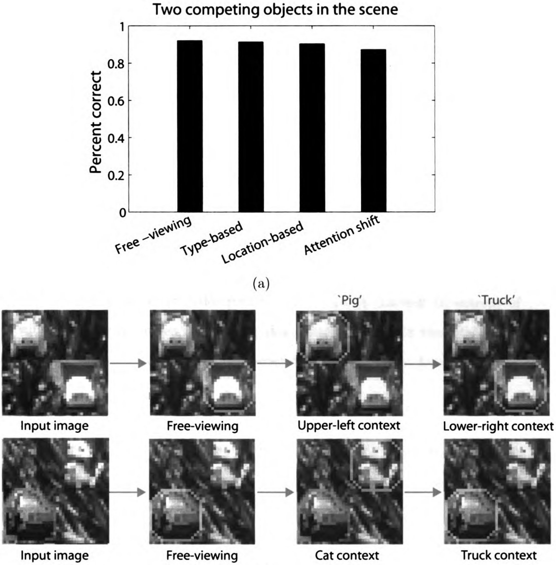
To simulate a shortage of neuronal resource relative to the input variability, we used a small network, five classes of objects, with images of a single size, and many different natural backgrounds. Both the training and testing sets used the same 5 object classes, but different background images. As there are only 3 V4 neurons at each location but 15 training object views, the WWN is 4/5 = 80% short of resource to memorize all the foreground objects. Each V4 neuron must deal with various misalignment between an object and its receptive field, simulating a more realistic resource situation. Location was tested in all 20 × 20 = 400 locations.

To see how a network does as it learns, we tested a single foreground in free-viewing mode throughout the learning time. As reported in Fig. 6.15(b), a network gave respectable performance after only the first round (epoch) of practice. After 5 epochs of practice, this network reached an average location error around 1.2 pixels and a correct disjoint classification rate over 95%.

6.3.2 Two Object Scenes

After training had progressed enough so the bottom-up performance with a single foreground was sufficient, we wished to investigate WWN-3's ability with two objects, and in top-down attention. We tested the above trained WWN-3 with two competing objects in each image, placed at four possible quadrants to avoid overlapping. We placed two different foregrounds in two of the four corners. There were 5 classes × 4 corners × 3 other corners (for the second foreground) = 60 combinations. WWN-3 first operated starting in free-viewing mode (no imposed motors), until it converged. If the type and location (within 5 pixels) matched one of the foregrounds, it was considered a success. Next, the type of the foreground that wasn't located was imposed at TM as an external goal, and WWN-3 operated in top-down searching mode to locate the other foreground. Next, WWN-3 would shift its attention back to the first object. Finally, the location of the foreground that wasn't identified in the first phase was imposed at LM as an external goal, and WWN-3 operated in top-down location-based mode to find the other foreground.

As shown in Fig. 6.15, the success rates for this network were 95% for the free-viewing test. The success rates were 96% when given type context to predict location and 90% when given location context to predict type. It successfully attended to the other object via an attention shift 83% of the time.



(a)

(b)

Figure 6.15: WWNs for the joint attention-recognition problem in bias-free and goal-based modes for two object scenes. (a) Performance when input contains two learned objects: bias-free, two types of imposed goals (top-down type-context for search and location-context for orient and detect), and shifting attention from one object to the other. (b) A few examples of operation over different modes by a trained WWN-3. "Context" means top-down goal is imposed. An octagon indicates the location and type action outputs. The octagon is the default receptive field.

6.3.3 Cross-Layer Connections

Callaway [18] discussed how the cortex might use alternate pathways through the pulvinar and thalamus so that earlier area can send information in a more direct way to later areas. These alternate pathways would allow the higher level areas to have higher resolution versions of the input signals that may have been significantly transformed in the cortico-cortical pathways going through many areas. This experiment tests an alternate pathway in WWN, but in the opposite direction.

The purpose of this section is to investigate the performance effects of direct top-down connections from the type-motor to V2. The direct-connection network maintains nearly "pure" type specificity in the earliest layer, leading to higher recognition accuracy for the limited set of foregrounds tested.

The following experiment is designed to test the prediction that (i) entropy of V2 will be greatly decreased through top-down connections from that motor and (ii) such low entropy can lead to higher recognition rates, given we have sufficiently large resource.

There were two network architecture types trained, as seen in Fig. 6.16. The first used top-down connections directly from TM to V2. The second architecture did not use the TM to V2 connections. V2 contained $20 \times 20 \times 3 = 1200$ neurons, PP and IT contained 20×20 neurons, there were 5 types and $20 \times 20 = 400$ location classes. Training occurred in the same way as before.

Every neuron had its type and location winning probabilities recorded through training. A simple way to measure how well the architecture would do with direct connections to TM and LM and a large amount of training experience is to take the winning V2 neuron's highest type probability as the output type, as indicated in Fig. 6.16 as the "V2 entropy classifier" block.

Comparisons between V2 entropy-based classification and the classification through IT and PP are shown in Tables 6.1 and 6.2 (for the disjoint test data), with the first

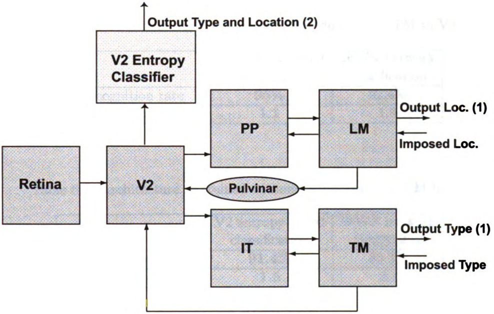


Figure 6.16: WWN-3 trains V2 through pulvinar location supervision and bottom-up based LCA. We added a new direct connection from TM so that V2 develops heightened type-specificity (even though it was already fairly type-specific). To test the coupled specificity of V2 representations, an alternate classifier, based on the winning neuron's entropy, was developed.

table showing the result for the architecture with direct top-down connections from TM to V2 used in training. The recognition rates and location error (measured in pixels) show that the architecture that used direct top-down connections is better overall. As expected, using the classification paths through IT and PP slightly decreased the performance. Average entropy is shown in Table 6.3. Architecture 1 led to a greatly reduced type-entropy (which was close to zero) in V2, as compared to architecture 2.

Table 6.1: Architecture 1: trained with top-down from TM to V2

	V2 entropy-based classification	WWN network classification
Recognition rate	**95%**	92.4%
Distance error (pixel)	**1.1**	1.9

Table 6.2: Architecture 2: trained without top-down from TM to V2

	V2 entropy-based classification	WWN network classification
Recognition rate	**91.4%**	89.4%
Distance error (pixel)	**1.5**	2.1

Table 6.3: Average entropy for both architectures, in bits

	Architecture 1	Architecture 2
V2 (what)	**0.05**	0.28
IT	0.16	0.18
V2 (where)	2.7	2.6
PP	2.4	2.2

This experiment shines light on a question of architecture selection for developmental recurrent networks. Should earlier areas have direct (cross-layer) connectivity to and from much later areas? Enabling entropy reduction from sensors to motors is probably a principle of development. But if the data is too complex, then it takes too much resource to reduce entropy enough within a single layer. It will take a large amount of representative resource to recognize a large number of objects without any shared features. Using top-down direct connections can increase the class-selectivity of the neurons, as shown above, but probably at the expense of efficient resource utilization.

6.4 Summary

The work here demonstrated that the WWN-3 is capable of bottom-up and top-down attention when two objects are in the scene. It uses internal synchronization through top-down connections to avoid the binding problem for this data. Either "where" or "what", to provide top-down context bias, as a goal or preference. Experiments using the disjoint foreground object subimages with general object contours reached an encouraging performance level by a limited size WWN-3.

6.5 Bibliographical Notes

The first version WWN-1 [48] operated on single foregrounds over random natural image backgrounds: type recognition given a location (top-down location based) and location finding given a type (top-down object based), where 5 locations were tested. The second version WWN-2 [47] additionally realized attention and recognition for single objects in natural backgrounds without supplying either position or type (free-viewing), and also used a more complex architecture. Further, all pixel locations were tested. The work reported here corresponds to the third version of WWN — WWN-3 [63]. It extends the prior versions WWN-1 and WWN-2 to deal with multiple objects in natural backgrounds, multiple views per class, non-square object contours, and disjoint testing foregrounds.

Chapter 7

Concluding Remarks

The main contribution of this work are the methods and theory about utilizing excitatory top-down connections in multilayer developmental networks. Top-down connections distribute positive bias, which can be exploited and controlled using this work's methods. There are two types of effects: spatial and temporal. **Spatial:** The topographic class grouping framework addresses the dual problems of feature extraction and automatic network wiring, by illustrating how and why top-down connections can cause biased (reduced-entropy) features and modular networks to emerge from incremental experience. **Temporal**: The firing of higher-layer units represents an abstract bias distribution. Top-down connections propagate this bias to lower layer units, and lower layer units feed back into the higher-layer units. It was shown that a clear interpretation of top-down connections in modular networks is as appropriate boosting of sub-features. Lateral inhibition, paired layers, and firing thresholds are effective to avoid incremental corruption of that temporal context over time. But in networks without modularity, this interpretation does not apply.

The theme of this work is cortex-inspired developmental vision. The design of our visual cortex was evolved to deal with the problems of many-objects representation, attention, and binding that need to be solved for a developmental vision system. Therefore, study of the cortex should lead to important insights about how to determine parameters and evaluate the artificial networks. Many of the cortical inspirations for these mechanisms and architectures are explained elsewhere in this work. To point out a few: 1. top-down connections are as numerous as bottom-up connections and nearly all layers that are connected are connected in a bidirectional way, and 2. Modularity is an important organizing principle. The role of recurrence at all layers of deep feature hierarchies and attention systems has not sufficiently been explored. This work is a step towards this goal.

Where-what networks are recurrent networks for attention and recognition. They use separate feature maps to represent location and type separately. It was shown how recurrent connectivity can enforce multilayer synchronization. This handles the binding problem without occlusions and transparency, without using combination neurons. When there are multiple objects in the scene, an unconstrained search for a single object's location and type has multiple solutions that can be correct with respect to the network but incorrect with respect to the world. From an unattended image, the network generates a plausible hypothesis along each path. By letting one of the unknowns become a constraint, the other is synchronized via top-down bias to the earlier layer followed by updated forward activity along the other path.

Experimental results are also key contributions. Some results are summarized here. In the LCA framework, it was shown how dual optimality greatly assists LCA's speed and precision of feature extraction, as compared to other Hebbian learning rules that use a manually tuned learning rate. And LCA's "CCI plasticity" showed long-term plasticity was possible even with such a fast convergence. Topographic class grouping emerged for handwritten digit recognition and recognition of objects from different 3D viewpoints; and the networks that learned with enabled top-down connections developed more effective compression than those not using them. They also developed modularity. Combining top-down connections with adaptive lateral-excitation led to a better compression compared to isotropic updating, but at the expense of smoothness and connector hubs. The modular networks developed were shown to use recurrence for significant performance boosts in object recognition and vehicle recognition (using local features) when the data was temporally ordered. Where-what networks successfully performed bottom-up and top-down attention when there were multiple objects in the scene.

The future of this work involves extending WWNs via a larger ventral pathway — a feature hierarchy. It would also be interesting to look into a motion pathway for sequence or trajectory learning in vision using WWNs. The associative information filling in effects of recurrent excitation will be examined. Finally, networks should be embedded into active agents, which will interact with and learn about the world.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] P.J. Antsaklis and A.N. Michel. Linear systems. 2006.

[2] G. Backer, B. Mertsching, and M. Bollmann. Data and model-driven gaze control for an active-vision system. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(12):1415–1429, 2001.

[3] R. Baillargeon. Object permanance in 3.5 and 4.5-month-old infants. *Developmental Psychology*, 23:655–664, 1987.

[4] C.I. Baker, C. Keysers, J. Jellema, B. Wicker, and D. I Perrett. Neuronal representation of disappearing and hidden objects in temporal cortex of macaque. *Exp. Brain Res.*, 140:375–381, 2001.

[5] H. Barlow. Redundancy reduction revisited. *Network: Computation in Neural Systems*, 12:241–253, 2001.

[6] HB Barlow, C. Blakemore, and JD Pettigrew. The neural mechanism of binocular depth discrimination. *The Journal of Physiology*, 193(2):327, 1967.

[7] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs fisherfaces: Recognition using class specific linear projection. 19(7):711–720, July 1997.

[8] A. J. Bell and T. J. Sejnowski. The 'independent components' of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

[9] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological review*, 94(2):115–147, 1987.

[10] I. Biederman and P.C. Gerhardstein. Recognizing depth-rotated objects: Evidence and conditions for three-dimensional viewpoint invariance. *Journal of Experimental Psychology: Human perception and performance*, 19(6):1162–1182, 1993.

[11] C. Blakemore and G. F. Cooper. Development of the brain depends on the visual environment. *Nature*, 228:477–478, Oct. 1970.

[12] J.-P. Braquelaire and L. Brun. Comparison and optimization of methods of color image quantization. *IEEE Transactions on Image Processing.*, 6(7):1048–1051, 1997.

[13] DE Broadbent. Perception and Comunication. 1958.

[14] J. Bullier. Hierarchies of cortical areas. In J.H. Kaas and C.E. Collins, editors, *The Primate Visual System*, pages 181–204. CRC Press, New York, 2004.

[15] E. Bullmore and O. Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186–198, 2009.

[16] D. V. Buonomano and U.R. Karmarkar. How do we tell time? *Neuroscientist*, 8:42–51, 2002.

[17] P. Buzas, K. Kovacs, A.S. Ferecsko, J.M.L. Budd, U.T. Eysel, and Z.F. Kisvarday. Model-based analysis of excitatory lateral connections in the visual cortex. *Journal of Comparative Neurology*, 499:861–881, 2006.

[18] E. M. Callaway. Local circuits in primary visual cortex of the macaque monkey. *Annu. Rev Neurosci*, 21:47–74, 1998.

[19] L.F. Chen, H.Y.M. Liao, M.T. Ko, J.C. Lin, and G.J. Yu. A new LDA-based face recognition system which can solve the small sample size problem. *Pattern recognition*, 33(10):1713–1726, 2000.

[20] Y-M Cheung and L. Law. Rival-model penalized self-organizing map. *IEEE Transactions on Neural Networks*, 18:289–295, 2007.

[21] BG Cumming and AJ Parker. Binocular neurons in V1 of awake monkeys are selective for absolute, not relative, disparity. *Journal of Neuroscience*, 19(13):5602, 1999.

[22] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. pages 428–441. Springer.

[23] P. Dayan and L.F. Abbott. *Theoretical neuroscience*. Citeseer, 2001.

[24] V.C. de Verdiere and J.L. Crowley. Visual recognition using local appearance. *Computer Vision (ECCV'98)*, page 640.

[25] G. Deco and E. T. Rolls. A neurodynamical cortical model of visual attention and invariant object recognition. *Vision Research*, 40:2845 – 2859, 2004.

[26] R. Desimone, TD Albright, CG Gross, and C. Bruce. Stimulus-selective properties of inferior temporal neurons in the macaque. *Journal of Neuroscience*, 4(8):2051–2062, 1984.

[27] R.. Desimone and J. Duncan. Neural mechanisms of selective visual attention. *Annual Review of Neuroscience*, 18:193–222, 1995.

[28] M. Dissanayake, P. Newman, S. Clark, HF Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[29] RJ Douglas, C. Koch, M. Mahowald, KA Martin, and HH Suarez. Recurrent excitation in neocortical circuits. *Science*, 269(5226):981, 1995.

[30] S. Edelman. *Representation and recognition in vision.* The MIT Press, 1999.

[31] J.L. Elman. Finding structure in time. *Cognitive Science: A Multidisciplinary Journal*, 14(2):179–211, 1990.

[32] D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1:1–47, 1991.

[33] D.J. Felleman and D.C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cereb. Cortex*, 1(11):1–47, 1991.

[34] M. Franzius, N. Wilbert, and L. Wiskott. Invariant object recognition with slow feature analysis. *Artificial Neural Networks-ICANN 2008*, pages 961–970.

[35] D.J. Freedman, M. Riesenhuber, T. Poggio, and E.K. Miller. A comparison of primate prefrontal and inferior temporal cortices during visual categorization. *Journal of Neuroscience*, 23(12):5235, 2003.

[36] K. Fukushima, S. Miyake, and T. Ito. Neocognitron: A neural network model for a mechanism of visual pattern recognition. 13(5):826–834, 1983.

[37] F. Gebali. Reducible Markov Chains. *Analysis of Computer and Communication Networks*, pages 1–32.

[38] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, New York, 1991.

[39] G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[40] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[41] JJ Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[42] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurons in the cat's striate cortex. *Journal of Physiology*, 148:574–591, 1959.

[43] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160(1):107–155, 1962.

[44] A. Hyvarinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.

[45] M. Ito and C. D. Gilbert. Attention modulates contextual influences in the primary visual cortex of alert monkeys. *Neuron*, 22:593–604, 1999.

[46] L. Itti and C. Koch. Computational modelling of visual attention. *Nat. Rev. Neurosci*, 2:194–203, 2001.

[47] Z. Ji and J. Weng. Where what network-2: A biologically inspired neural network for concurrent visual attention and recognition. In *IEEE World Congress on Computational Intelligence*, Spain, 2010.

[48] Z. Ji, J. Weng, and D. Prokhorov. Where-what network 1: "where" and "what" assist each other through top-down connections. In *Proc 7th Int'l Conf on Development and Learning*, Monterey, CA, August 9-12 2008.

[49] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[50] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, editors. *Principles of Neural Science*. McGraw-Hill, New York, 4th edition, 2000.

[51] N. Kanwisher. Repetition blindness and illusory conjunctions: Errors in binding visual types with visual tokens. *Journal of Experimental Psychology: Human Perception and Performance*, 17(2):404–421, 1991.

[52] C. Koch. *The Quest for Consciousness: A Neurobiological Approach*. Roberts and Company Publishers, Englewood, Colorado, 2004.

[53] C. Koch and S. Ullman. Shifts in selective visual attention: Towards the underlying neural circuitry. *Human Neurobiology*, 4:219–227, 1985.

[54] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 2nd edition, 1997.

[55] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 3rd edition, 2001.

[56] S. L. Pallas L. von Melchner and M. Sur. Visual behavior mediated by retinal projections directed to the auditory pathway. *Nature*, 404:871–876, 2000.

[57] V.A.F. Lamme. Blindsight: the role of feedforward and feedback corticocortical connections. *Acta Psychol*, 107:209–228, 2001.

[58] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[59] Y. LeCun, F.J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[60] D Lee and S Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[61] E. L. Lehmann. *Theory of Point Estimation*. John Wiley and Sons, Inc., New York, 1983.

[62] M. Luciw and J. Weng. Laterally connected lobe component analysis: precision and topography. In *Proc. 9th International Conf. on Development and Learning (ICDL '09)*, Shanghai, China, June 5 - 7 2009.

[63] M. Luciw and J. Weng. Where what network-3: Developmental top-down attention with multiple meaningful foregrounds. In *Proc. IEEE World Congress on Computational Intelligence*, Barcelona, Spain, 2010.

[64] M. D. Luciw and J. Weng. Topographic class grouping with applications to 3D object recognition. In *Proc. International Joint Conference on Neural Networks*, Hong Kong, June 1-6 2008.

[65] Matthew D. Luciw. Multilayer in-place learning for autonomous mental development. Master's thesis, MIchigan State University, 2006.

[66] J. Lucke and M. Sahani. Maximal causes for non-linear component extraction. *The Journal of Machine Learning Research*, 9:1227–1267, 2008.

[67] D.G. Mackay. Aspects of the theory of comprehension, memory and attention. *The Quarterly Journal of Experimental Psychology*, 25(1):22–40, 1973.

[68] P.E. Maldonado, I. Godecke, C.M. Gray, and T. Bonhoeffer. Orientation selectivity in pinwheel centers in cat striate cortex. *Science*, 276:1551–1555, 1997.

[69] Y. MarcAurelio Ranzato, L. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. *Advances in Neural Information Processing Systems*, 20:1185–1192.

[70] D. Marr. *Vision.* Freeman, New York, 1982.

[71] J.H. Maunsell and D.C. Van Essen. The connections of the middle temporal visual area (mt) and their relationship to a cortical hierachy in the macaque monkey. *J. Neuroscience*, 3(12):2563–2586, 1983.

[72] B.A. McGuire, C.D. Gilbert, P.K. Rivlin, and T.N. Wiesel. Targets of horizontal connections in macaque primary visual cortex. *J Comp Neurol*, 305:370–392, 1991.

[73] R. Miikkulainen, J. A. Bednar, Y. Choe, and J. Sirosh. *Computational Maps in the Visual Cortex.* Springer, Berlin, 2005.

[74] M. Mishkin, L.G. Ungerleider, and K.A. Macko. Object vision and spatial vision: Two cortical pathways. *Philosophy and the neurosciences: a reader*, page 199, 2001.

[75] M.C. Mozer, M.H. Wilder, and D. Baldwin. A Unified Theory of Exogenous and Endogenous Attentional Control. *Department of Computer Science and Institute of Cognitive Science University of Colorado, Boulder, CO 80309*, 430, 2007.

[76] H. Murase and S.K. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

[77] E. Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

[78] BA Olshausen, CH Anderson, and DC Van Essen. A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *Journal of Neuroscience*, 13(11):4700, 1993.

[79] B.A. Olshausen and D.J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision research*, 37(23):3311–3325, 1997.

[80] B. A. Olshaushen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, June 13 1996.

[81] S.M. Pan and K.S. Cheng. An evolution-based tabu search approach to codebook design. *Pattern Recognition*, 40(2):476–491, 2007.

[82] A. Pascual-Leone and V. Walsh. Fast back projections from the motion to the primary visual area necessary for visual awareness. *Science*, 292:510–512, 2001.

[83] A. Pasupathy and C.E. Connor. Shape representation in area V4: position-specific tuning for boundary conformation. *Journal of Neurophysiology*, 86(5):2505, 2001.

[84] J. Piaget. *The Construction of Reality in the Child.* Basic Books, New York, 1954.

[85] R.D. Raizada and S. Grossberg. Towards a theory of the laminar architecture of cerebral cortex: computational clues from the visual system. *Cereb Cortex*, 13:100C113, 2003.

[86] M. Riesenhuber and T. Poggio. Models of object recognition. *Nature Neuroscience*, 3:1199–1204, 2000.

[87] I. Rock and J. DiVita. A case of viewer-centered object perception. *Cognitive Psychology*, 19(2):280–293, 1987.

[88] H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. page 1746, 2000.

[89] TJ Sejnowski. Predictive sequence learning in recurrent neocortical circuits. *Advances in Neural Information Processing Systems 12*, page 164, 2000.

[90] R.N. Shepard and J. Metzler. Mental rotation of three-dimensional objects. *Science*, 171(3972):701, 1971.

[91] E. Simoncelli and B. Olshausen. Natural image statistics and neural representation. *Annu. Rev. Neurosci.*, 24:1193 - 1216, 2001.

[92] L.C. Sincich and J.C. Horton. The circuitry of V1 and V2: integration of color, form, and motion. 2005.

[93] Y.F. Sit and R. Miikkulainen. Self-organization of hierarchical visual maps with feedback connections. *Neurocomputing*, 69:1309–1312, 2006.

[94] M. Solgi and J. Weng. Developmental Stereo: Emergence of Disparity Preference in Models of the Visual Cortex. *IEEE Trans. on Autonomous Mental Development*, 1(4):238–252, 2009.

[95] C. Southan. Has the yo-yo stopped? An assessment of human protein-coding gene number. *Proteomics*, 4(6):1712–1726, 2004.

[96] M. Sur, A. Angelucci, and J. Sharm. Rewiring cortex: The role of patterned activity in development and plasticity of neocortical circuits. *Journal of Neurobiology*, 41:33–43, 1999.

[97] D. L. Swets and J. Weng. Using discriminant eigenfeatures for image retrieval. 18(8):831–836, 1996.

[98] K. Tanaka. Inferotemporal cortex and object vision. *Annual Reviews of Neuroscience*, 19:109–139, 1996.

[99] M.J. Tarr and H.H. Bulthoff. Is human object recognition better described by geon structural descriptions or by multiple views? Comment on Biederman and Gerhardstein (1993). *Journal of Experimental Psychology-Human Perception and Performance*, 21(6):1494–1505, 1995.

[100] J.G. Taylor. The CODAM model of Attention and Consciousness. pages 292–297, 2003.

[101] JG Taylor. CODAM: A neural network model of consciousness. *Neural Networks*, 20(9):983–992, 2007.

[102] J.M. Hupe *et al.* Cortical feedback improves discrimination between figure and background by v1, v2 and v3 neurons. *Nature*, 394:784–787, 1998.

[103] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *nature*, 381(6582):520–522, 1996.

[104] H. Tomita, M. Ohbayashi, K. Nakahara, I. Hasegawa, and Y. Miyashita. Top-down signal from prefrontal cortex in executive control of memory retrieval. *Nature*, 401(6754):699–703, 1999.

[105] R.B.H. Tootell, K.J. Devaney, J.C. Young, R. Rajimehr G. Postelnicu, and L.G. Ungerleider. fmri mapping of a mophed continum of 3d shapes within inferior temporal cortex. *Proc Natl Acad Sci USA*, 105:3605–3609, 2008.

[106] A. Treisman. Monitoring and storage of irrelevant messages in selective attention1. *Journal of Verbal Learning and Verbal Behavior*, 3(6):449–459, 1964.

[107] A. Treisman. Solutions to the binding problem: review progress through controversy summary and convergence. *Neuron*, 24:105–110, 1999.

[108] A. Treisman and H. Schmidt. Illusory conjunctions in the perception of objects. *Cognitive Psychology*, 14(1):107–141, 1982.

[109] A.M. Treisman and G. Gelade. A feature-integration theory of attention. *Cognitive psychology*, 12(1):97–136, 1980.

[110] D.Y. Ts'o, C.D. Gilbert, and T.N. Wiesel. Relationships between horizontal interactions and functional architecture in cat striate cortex as revealed by cross-correlation analysis. *J Neurosci*, 6:1160–1170, 1986.

[111] J. K. Tsotsos, S. M. Culhane, W. Y. K. Wai, Y. Lai, N. Davis, and F. Nuflo. Modeling visual attention via selective tuning. *Artificial Intelligence*, 78:507–545, 1995.

[112] J.K. Tsotsos, J. Konstantine, and University of Toronto. Dept. of Computer Science. The complexity of perceptual search tasks. 1989.

[113] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[114] Alhoniemi E Vesanto J, Himberg J and Parhankangas J. Som toolbox for matlab 5. Technical Report A57, Helsinki University of Technology: Finland, 2000.

[115] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. volume 1, page 511, 2001.

[116] C. Von Der Malsburg. The what and why of binding: review the modelers perspective. *Neuron*, 24:95–104, 1999.

[117] J. Weng. Task muddiness, intelligence metrics, and the necessity of autonomous mental development. *Minds and Machines*, 19(1):93–115, 2009.

[118] J. Weng, N. Ahuja, and T. S. Huang. Cresceptron: a self-organizing neural network which grows adaptively. In *Proc. Int'l Joint Conference on Neural Networks*, volume 1, pages 576–581, Baltimore, Maryland, 1992.

[119] J. Weng, N. Ahuja, and T. S. Huang. Learning recognition and segmentation using the Cresceptron. 25(2):109–143, Nov. 1997.

[120] J. Weng, H. Lu, T. Luwang, and X. Xue. A multilayer in-place learning network for development of general invariances. *International Journal of Humanoid Robotics*, 4(2), 2007.

[121] J. Weng, H. Lu, T. Luwang, and X. Xue. Multilayer in-place learning networks for modeling functional layers in the laminar cortex. *Neural Networks*, 2008.

[122] J. Weng and M. Luciw. Dually-optimal neuronal layers: Lobe component analysis. *IEEE Transactions on Autonomous Mental Development*, 1(1), 2009.

[123] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. *Science*, 291(5504):599–600, 2001.

[124] J. Weng and N. Zhang. Optimal in-place learning and the lobe component analysis. In *Proc. World Congress on Computational Intelligence*, Vancouver, Canada, July 16-21 2006.

[125] J. Weng, Y. Zhang, and W. Hwang. Candid covariance-free incremental principal component analysis. 25(8):1034–1040, 2003.

[126] PJ Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[127] P.J. Werbos. *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting.* Wiley-Interscience, 1994.

[128] X. Wu and K. Zhang. A better tree-structured vector quantizer. In *Data Compression Conference, 1991. DCC'91.*, pages 392–401, 1991.

[129] Y. Zhang, J. Weng, and W. Hwang. Auditory learning: A developmental method. *IEEE Transactions on Neural Networks*, 16(3):601–616, 2005.