A Computational Dynamical Model of Human Visual Cortex for Visual Search and Feature-based Attention

David Graham Harrison

Submitted in accordance with the requirements for the degree of
Doctor of Philosophy

The University of Leeds
School of Computing

September, 2012

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly-authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

The following articles have been published during the period of study for this PhD:

D. G Harrison and M. de Kamps. "A Dynamical Neural Simulation of Feature-based Attention and Binding in a Recurrent Model of the Ventral Stream", *Proceedings of the 12^{t}h Neural Computation and Psychology Workshop*, (2010), 40–57. My contributions: The work in this paper is all my own. Other author contributions: M. de Kamps provided supervision and corrections. Chapters based on this work: Chapters 3 and 4.

D. G Harrison and M. de Kamps. "A Dynamical Model of Feature-Based Attention with Strong Lateral Inhibition to Resolve Competition Among Candidate Feature Locations", *Proceedings of the AISB 2011 Symposium on Architectures for Active Vision*, 2011. My contributions: The work in this paper is all my own. Other author contributions: M. de Kamps provided supervision and corrections. Chapters based on this work: Chapters 3 and 4.

Acknowledgements

This PhD was funded by the White Rose University Consortium. I gratefully acknowledge Jim Austin, Netta Cohen, Marc de Kamps. Kevin Gurney, Becky Naylor, Simon O'Keefe, Tom Stafford and David Yates, who have all supported this PhD through many discussions at White Rose 'Active Vision Network' meetings.

I am hugely grateful to Marc de Kamps for his supervision throughout my PhD. Our meetings were frequently informal, which fostered an open environment to discuss many ideas freely.

Finally, I would like to thank Elaine Duffin, who donated significant amounts of her time to proof multiple drafts of this thesis. Her input has been hugely valuable and improved this final copy greatly.

Abstract

Visual attention can be deployed to locations within the visual array (spatial attention), to individual features such as colour and form (feature-based attention), or to entire objects (object-based attention). Objects are composed of features to form a perceived 'whole'. This compositional object representation reduces the storage demands by avoiding the need to store every type of object experienced. However, this approach exposes a problem of binding these constituent features (e.g. form and colour) into objects. The problem is made explicit in the higher areas of the ventral stream as information about a feature's location is absent. For feature-based attention and search, activations flow from the inferotemporal cortex to primary visual cortex without spatial cues from the dorsal stream, therefore the neural effect is applied to all locations across the visual field [79, 60, 7, 52].

My research hypothesis is that biased competition occurs independently for each cued feature, and is implemented by lateral inhibition between a feedforward and a feedback network through a cortical micro-circuit architecture. The local competition for each feature can be combined in the dorsal stream via spatial congruence to implement a secondary spatial attention mechanism, and in early visual areas to bind together the distributed featural representation of a target object.

Contents

List of Tables

List of Figures

Glossary

AIT Anterior Inferotemporal.

ANN Artificial Neural Network.

CLAM Closed Loop Attention Model.

CPU Central Processing Unit.

DBN Deep-belief Network.

DIANNA Dynamical Interacting Artificial Neural Network Application.

EEG Electroencephalogram.

FEF Frontal Eye Fields.

fMRI Functional Magnetic Resonance Imaging.

IOR Inhibition of Return.

LGN Lateral Geniculate Nucleus.

LIF Leaky-integrate-and-fire.

LIP Lateral Intraparietal.

MIIND Multiple Instantiations of Interacting Neural Dynamics.

MST Medial Superior Temporal.

MT Middle Temporal.

PIT Posterior Inferotemporal.

PNG Portable Network Graphics.

PP Posterior Parietal.

RNG Random Number Generator.

SOM Self-Organizing Map.

XML Extensible Markup Language.

XOR Exclusive OR.

Chapter 1

Introduction

The amount of information reaching the retina is considerably more than the brain can process [73], yet organisms still need to respond to their environment accurately and rapidly to avoid threats, find and react to prey, or find a mate. Organisms therefore require mechanisms to reduce the retinal input to a volume of data the brain is capable of processing. Visual attention is a mechanism common among higher animals (particularly mammals) to reduce detailed processing of the retinal input to a smaller area of input which is considered salient to the organism's current task. A natural problem arises with this approach: which part of the visual scene is the most salient in order to direct further cortical processing? For an organism to ensure its survival it must be able to determine which portion of the visual field to consider based upon its current situation and needs, and be able to switch this "spotlight" [12] as new stimuli appear.

The brain is a highly complex structure, with $86.1 \times 10^9 \pm 10\%$ neurons making 10^3 inter-connections with other neurons, with 19% of these neurons in the cerebral cortex [1]. The cortical neurons are organized into functional structures at various levels, from individual neurons to brain areas determined to be responsible for particular functions. Cortical visual structures are located dorsally, with more complex transformations of the stimulus driven neural activity occurring in anterior brain areas. Furthermore, two streams of visual processing have been determined, which interact to allow visual processing to be directed towards a salient location within the visual scene.

Understanding the structure of the visual brain, those areas of the brain concerned

with vision, is complicated by the multiple scales on which the brain performs its tasks. The scales range from the large number of tiny interactions between connected neurons and the distributed and noisy interactions that occur between neural populations at both the micro and macro scales (micro-circuits and brain areas respectively). The variety of scales makes the study of cortical vision at any one scale insufficient, so vision scientists create hypotheses and models of how the brain performs these functions. A model should allow predictions of how the brain will respond to particular stimuli, which can then be tested against observation in psychophysical and biophysical studies, to evaluate the model's plausibility.

This thesis presents a computational model of visual attention at multiple levels of scale and interaction. The model comprises layers of neurons representing the different cortical areas and their functions as composites of neural circuits. Using a dynamical model allows the time course of these neural interactions to be studied and compared to observed brain dynamics through areal mean field techniques, such as Electroencephalogram (EEG) and Functional Magnetic Resonance Imaging (fMRI), which record activity from populations of neurons, not individual cells.

1.1 Structural organization of the Visual Brain

Before presenting the model a discussion of the organization of the brain in terms of visual processing is presented. While approximately 30 cortical areas are implicated in visual processing [24], only the areas featured in the model of visual attention presented in this thesis are described, and depicted in figure 1.1.

1.1.1 The Main Cortical Pathways involved in Cortical Vision

Ungerleider and Mishkin [74] showed two main visual pathways in the brain: the ventral stream, passing from the occipital cortex ventrally to inferotemporal areas, and a dorsal stream from occipital cortex to posterior-parietal areas via middle temporal areas. A discussion of the visual cortical areas these pathways subsume follows this section.

Ungerleider and Mishkin suggested the role of the ventral stream is the perception of objects and form, and consequently termed it the 'what' pathway, while the

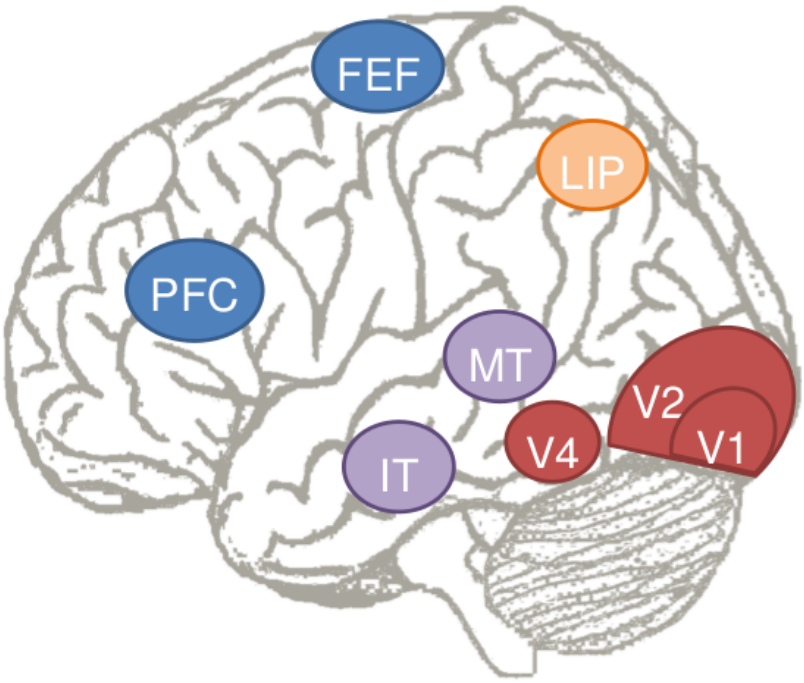


Figure 1.1: Approximate locations of visually related cortical areas.

dorsal stream, being concerned with spatial information, was termed the 'where' pathway [74, 46]. However, Milner and Goodale [30] (and Goodale [28, 29]) posit the dorsal stream is involved in visually mediated action. In both interpretations the ventral stream is primarily concerned with visual perception, while the dorsal stream is concerned with spatial characteristics of vision, such as location, spatial relationships between objects and motion.

1.1.2 Occipital Cortex

1.1.2.1 Primary Visual Cortex (V1)

The primary visual cortex, also known as V1, lies beneath the occipital bone at the rear of the skull. This area receives incoming neural input from the basal ganglia, and principally the Lateral Geniculate Nucleus (LGN), which in turn receives its neural input from retinal ganglion cells. Area V1 has been extensively studied [57, c. 2], and is the most spatially faithful brain area in terms of mapping to retinotopic locations. A seminal study of neural responses to given stimuli [33] by Hubel and

Wiesel showed neurons in V1 are strongly tuned to lines of orientation, with the highest spike rates elicited at the preferred line orientation and almost zero spikes for stimuli in the orthogonal anti-preferred orientations. This study showed that V1 neurons respond to local neural activity soon after presentation of stimuli, but [38] demonstrated V1 neurons respond to more global properties of the visual scene after 100ms, due to interactions from higher brain areas.

Hubel and Wiesel [33] proposed the organization of V1 as columns of neurons, with each column sensitive to lines of specific orientation. This columnar organization is found throughout the cortex, but Hubel and Wiesel were the first to postulate a direct link between this columnar organization and function in V1 (see 1.1.5 for discussion of cortical columns).

1.1.2.2 V2

Area V2 receives strong input from V1, and outputs stimulus driven activity both to the ventral stream areas of V3 (not shown in figure 1.1) and V4 and dorsal stream area of V3 and V5/MT, and strong feedback connectivity to V1 [57, c. 3]. It is less strongly orientation selective than V1, but codes for more complex conjunctions of features [57, c. 3].

1.1.2.3 V4

Visual area V4 is sensitive to both colour and orientation [89], and shows the strongest degree of attentional modulation of all visual areas: deployment of attention can alter the firing rate of neurons by as much as 20% [47]. Similar to V2, form features are complex conjunctions of features from the predecessor layer of the ventral stream.

1.1.2.4 Middle Temporal (V5)

Area MT/V5 has a role in visual perception of motion. It is involved in the computation of whole object velocity and the , as its neurons are tuned to both direction and speed of motion [42]. Its principle inputs are as few as five synapses from the photoreceptors [6], and it provides strong retinotopy. This area is one of the main

inputs to the dorsal stream [74, 43] including Medial Superior Temporal (MST) and Lateral Intraparietal (LIP), and projects to the Frontal Eye Fields (FEF) for the preparation of eye movements.

1.1.3 Inferotemporal Area

The inferotemporal area responds to complex visual stimuli, such as faces [21]. The models presented in this thesis divide this region into two visual areas: the Posterior Inferotemporal (PIT) and Anterior Inferotemporal (AIT) areas.

1.1.4 Other Visually Related Cortical Areas

1.1.4.1 Lateral Intraparietal (LIP)

Area Lateral Intraparietal (LIP) is located in the posterior parietal cortex, in the dorsal stream. This area is involved with visual spatial attention, with LIP neurons showing strong retinotopy [56]. LIP projects spatial information to the FEF.

1.1.4.2 Frontal Eye Fields (FEF)

The Frontal Eye Fields (FEF) is located in the prefrontal cortex, and is important in the generation of voluntary saccades [63]. The models presented in this thesis do not include this area.

1.1.5 Columnar Organization of the Cortex

The cortex is not simply a two dimensional structure, but is comprised of cortical columns [51]. The cortical columns are composed of six layers, numbered from the superficial layer 1, to the deepest layer 6. The relative thickness of these layers varies by region throughout the cortex. While little is known of the functional role of the superficial layers 1 and 2, layer 4 receives thalamic input, and layer 6 provides thalamic output [26, 27, 4]. Inter-columnar connections originating from layers 3 and 6 to layer 4 are termed ‘feed-forward projections’, while those which terminate in layers other than layer 4 are termed ‘feedback projections’ [4]. Within

a column there is significant inter-layer connectivity, with neurons projecting axons outside of the cortical column also connected to neurons within the same column [4].

While Mountcastle [51] postulated cortical columns as the functional unit of cortical computation, cortical columns with differing functions within the same receptive field (such as detecting lines of differing orientation and ocular dominance [33]), are termed cortical modules [51] or hypercolumns [27, 9].

1.2 Visual Attention

The amount of visual information entering the eye would overwhelm the brain's visual processing capability if the entire retinal input is processed equally at all locations [65]. In order to perceive visual objects in detail, the object of interest is brought into focus on the fovea. The mechanism by which we designate what is of interest is described as visual attention. The ability to visually attend to an object is so crucial to visual perception it has been argued that "to see is to attend" [88].

The mechanism of attention and how it may be captured or deployed is not well understood, in part due to technical difficulties in studying the brain and the complexity of the brain itself, despite modern technological developments in biophysics, such as fMRI which has provided a means for studying cortical activity in unanaesthetized subjects. However, fMRI traces have a poor spatial resolution of $3mm^3$ (approximately 7 million neurons) and a temporal latency due to measuring post activity oxygen absorption from the blood [8]. Other forms of study include behavioural experiments (psychophysics) on humans providing an indirect view of the brain, and animal experiments using techniques such as single electrode recording. These techniques each have their problems for investigating brain function so computer modelling presents a powerful technique to determine the function of visual cortex: a model can be tested for correctness by comparing against the vast amount of experimental data gathered through psychophysical and single electrode recording studies. Evaluation against these studies is difficult due to the large number of phenomena observed, some of which are contradictory. Also, fMRI detects different activity to single electrode recordings as each voxel only provides a population average and it is consequently difficult to reconcile with single electrode recordings as the effect on individual neurons, or the number of neurons respond-

ing, cannot be determined.

Three types of visual attention have been described: attention may be deployed to a location (spatial attention), an individual object (object-based attention), or to a collection of features (feature-based attention).

1.2.1 Spatial Attention

Visual attention can be deployed to locations in the visual field to increase neural sensitivity for neurons whose receptive fields are enclosed within the attended location. The neural effect of spatial attention is to increase the effective contrast of stimuli within the locus of attention [80, 55] and to reduce the response of neurons sensitive to unattended locations [72]. Treue and Martinez-Trujillo describe this process as the contrast gain model of spatial attention [72].

After visual search has located candidate locations for the sought object, spatial attention can then be applied to that location to further discern the candidate as the search target. Motter [48] describes spatial attention in these situations as shrinking the receptive field around the attended to object.

In order for spatial attention to be applied, the location of interest must first be selected. If the location is unknown, a visual search must be performed to determine locations from known properties of the object before spatial attention can be engaged. Neural mechanisms to determine the location for spatial attention have been described in the literature, such as saliency [34] or priority maps [5]. Priority maps, like saliency maps, code for a location of interest from visual stimuli, but include top-down influences in addition to bottom-up. The model presented in this thesis generates spatial saliency maps through the interaction of neural activity in top-down and bottom-up visual pathways. Influence of top-down flow is necessary to sustain output to the dorsal stream once a location for attention has been determined. See [87] for a review of visual search.

1.2.2 Feature-based Attention

Feature-based attention describes the deployment of attention to known properties of the visual scene. These properties are simple features such as colour, orientation and direction of motion [71]. Feature-based attention enhances the response of

neurons which code for the attended to feature [44]. Feature-based attention is used to detect the presence of the features in the visual scene into a retinotopic map, which is then used to resolve the location of those detected features. As the location of these features is not know prior to the onset of feature-based attention, the feature templates must act across the visual field [75, 7, 44, 65, 8, 90].

The top-down feature template interacts with bottom-up activity in two ways. In the feature-similarity gain principle [41] the sensitivity of neurons which code for the presence of the attended feature is enhanced, whilst the activity of neurons which do not code for the attended feature is suppressed.

This thesis presents a computational model of visual search which achieves a similar effect using feature-based attention (see chapter 4). The model employs lateral inhibition of cortical feature-binding circuits when those circuits receptive fields contain non-matching features to an endogenously initiated attentional template. The lateral inhibition from non-matching circuits subdues activity in neighbouring populations, effectively removing external stimuli in their receptive fields from further visual processing. Regions with few mismatches between the attentional template and stimulus driven activity project excitatory activity to a separate cortical area in the dorsal stream (for example, LIP), where the saccade necessary to foveate the object may be generated.

1.2.3 Object-based Attention

Object-based attention may be described as an example of feature-based attention, as evidence [53] suggests that object representations are composed of distributed sub-object, feature building blocks bound together as the neural object representation as needed, or determined, by the visual stimulus. These features are bound to each other through spatial relationships, such that attending to any component feature of an object causes selection of all other object features [62]. This compositional object representation reduces the storage demands by avoiding the need to store every type of object experienced (see section 1.7). The effect of object-based attention has been described as a shrinking of the receptive fields of neurons around the attended object [47].

1.3 The Biased Competition Model

In the biased competition model of feature-based attention [22], stimuli within a feature-selective neuron's receptive field compete for neural representation. When only a preferred stimulus is present within the receptive field and matching feature-based attention is applied, the response of the neuron is maximal. When an additional non-matching stimuli occurs in the receptive field the competition is biased in favour of those stimuli which match the coded for feature, causing a suppressed response to non-matching stimuli and increased response to the matching stimuli. When attention is applied to the non-preferred stimulus, the response of the neuron is suppressed, but still greater than the response of the neuron when the preferred stimulus is absent.

This thesis presents a model implementing biased competition through layer-local lateral inhibition in the ventral stream independently for each cued feature. The biased competition is initiated by a top-down attentional template which acts across all areas of the visual scene. The independent local competition for each feature is combined in early visual areas and activates a spatial attention mechanism in the dorsal stream via spatial congruence.

1.4 Feature-Similarity Gain versus Feature Matching

Treue and Martinez-Trujillo [72] performed experiments on macaque neurons in area Middle Temporal (MT), and measured modulation of neuronal responses when attention was directed to locations within the receptive field of these neurons. This modulation was multiplicative only, and did not affect the response of the neuron to any particular feature. This demonstrates the effect of spatial attention is independent of the features presented [45, 64] (although the strength of the neuronal modulation did vary with the contrast of the stimulus at the attended location). However, measurements of neurons in area MT, which are sensitive to direction of motion, showed variation in the neurons' tuning curves when attention to an object moving in specific direction was deployed, regardless of whether that object lay within a neuron's receptive field. When attention was directed to a feature moving in the preferred direction of a neuron, responses were higher than when attention was directed to the anti-preferred direction of motion. From this study

Treue and Martinez-Trujillo proposed the 'feature-similarity gain model', in which the attentional state affects the neuronal response depending on the similarity of the attended stimulus and a neuron's preferred stimulus, in all neurons sensitive to the attended stimulus, across the visual field.

An alternative hypothesis of attentional modulation of neuronal responses is described by Motter [49, 50] as a correlation between how well a stimulus within a neuron's receptive field matches the attended stimulus, even when the attended stimulus lies outside of the recorded neuron's receptive field. This description of neural modulation does not specify a neuron having a preferential feature to which it responds, and suggests modulation of a neuron's sensitivity to a given stimulus in the presence of attention. This effect is termed 'feature-matching' by Treue and Martinez-Trujillo [72].

In their study [72], Treue and Martinez-Trujillo found no evidence supporting the 'feature-matching' hypothesis of attention. However, their experimental stimuli consisted of groups of dots moving in two (preferred and anti-preferred) directions. Such simple movement is unlikely to require complex neural transformations of stimulus evoked neural activity, whereas the detection of form features is a more complex task, as simple features detected at lower visual areas are transformed into object representations in AIT through interactions in the intervening ventral stream layers.

The existence of the feature-similarity gain principle and the biased competition model as mechanisms for feature-based attention are not mutually exclusive mechanisms. Rather, it has been argued that the feature-similarity gain principle predicts the biased competition model [7]. Treisman and Gelade [71] detail a set of basic features (such as orientation, colour and direction of motion) to which neurons are sensitive. When attention is directed to one of these fundamental features, the activity of neurons sensitive to that feature are modulated such that the response to stimuli that match the coded for feature is increased, and the response of other neurons which are not selective to the attended feature are suppressed. The selectivity of feature-sensitive neurons is a continuum: a neuron sensitive to horizontal lines will see a response gain to the presence of a horizontal line within its receptive field, a depressed response for the presence of a vertical line and a diminished response to intermediate orientations. The level of modulation depends on the similarity between the coded for feature and the stimulus [41, 44]. Sàenz *et al* [60] present

results consistent with the feature similarity gain model [72] while showing that the attentional effects are dependent on the presence of competing stimuli.

1.5 Attentional Capture

Visual attention may be deployed voluntarily, as in visual search or Posner (delayed-match-to-sample) cuing paradigms, but may also be initiated from external stimuli. Sudden changes in visual input, such as an unexpected flash, will create a neural activity in populations coding for the location of the external change. This effect of breaking into conscious perception of a salient but irrelevant stimulus at a non-attended location is an example of 'implicit attentional capture' [67]. When attentional mechanisms are not engaged, attention may be briefly captured by this new stimulus, termed 'explicit attentional capture' [67]. Once the cause of stimulus has been ascertained, the stimulus may be ignored, with a concomitant reduction in the representative LIP activity, or actively attended, maintaining the location activity in LIP.

1.6 Visual Search

Visual attention can be applied to objects, features or locations, and may be either captured by incoming stimuli or voluntarily deployed by endogenous cortical activity. For attention to be applied to an object, the object must be recognized from its two dimensional retinal representation under varying conditions of lighting, rotation and scale. This raises interesting questions in how objects are efficiently stored in the brain such that recognition can occur invariant of the actual presentation (see section 1.7. When searching a cluttered visual scene, attention is deployed to a number of spatial positions sequentially, many times per second, until the target object is found through recognition, or the search is abandoned [71, 85, 70]. Visual search is often guided by information outside the focus of attention, and can be attracted to particular objects with such vigour that the object appears to jump out from the background, such that they appear more salient. This shows the deployment of attention can be stimulus driven as well as cortically driven.

In navigating the visual environment, people must attend to some objects and ig-

nore others. Guiding attention to the relevant object is easy in some instances, as when picking out a red ball from a collection of blue balls, but more difficult in others, as when attempting to find a suitable piece of a jigsaw puzzle. To quantify how attention is used in these perceptual tasks, behavioural research has relied extensively on analyses of visual search [21], in which observers attempt to find a predefined target item in a display of accompanying non-target (distractor) items. The measure of interest is often the change in reaction time (RT) or accuracy for detecting the target, in relation to an increasing number of display items (display size). As illustrated in panel (b) of Figure 1.2, the search target (e.g., a red circle) may be a featural singleton, defined as an item that differs from all of the distractors in a particular feature (e.g., colour, shape, or size). Detection of a singleton target is highly efficient, and RT is relatively unaffected by the number of distractor items [71]. In contrast, when the target and distractors share features (Figure 1.2, panels (a) & (c)), search is less efficient, and target detection requires additional time as display size increases.

Current theories of visual attention characterize the difference between panels (a) and (b) in Figure 1.2 in terms of a continuum of search efficiency, rather than as categorically different forms or stages of information processing [86]. Treisman and Gelade [71] and Itti and Koch [34] argue that search is highly efficient (as in panel (b)) when driven entirely in a bottom up manner: that is, by salient differences among the features of the display items. Conversely, Hochstein and Ahissar [32] argue that the global saliency of such singletons amongst distractors originates from feature categories maintained in higher visual areas, requiring a top-down mechanism for 'pop-out' to occur. Van der Velde and de Kamps [77] present a model of global visual saliency in which the bottom-up representation of the distractor objects would overwhelm the neural representation of the singleton without the interaction of a top-down attentional template. The van der Velde and de Kamps model may still be described as highly efficient, as search is performed by parallel processes.

In highly efficient search, the observer has the impression that the target pops out of the display, capturing attention automatically. Successful performance in more difficult search tasks (as in panel (a)) relies on top-down processing: that is, the observer's knowledge of the target and how it differs from the distractors. In this type of inefficient or difficult search, there is typically a direct increase in RT as a function of increasing display size, and thus the slope of the line of RT plotted

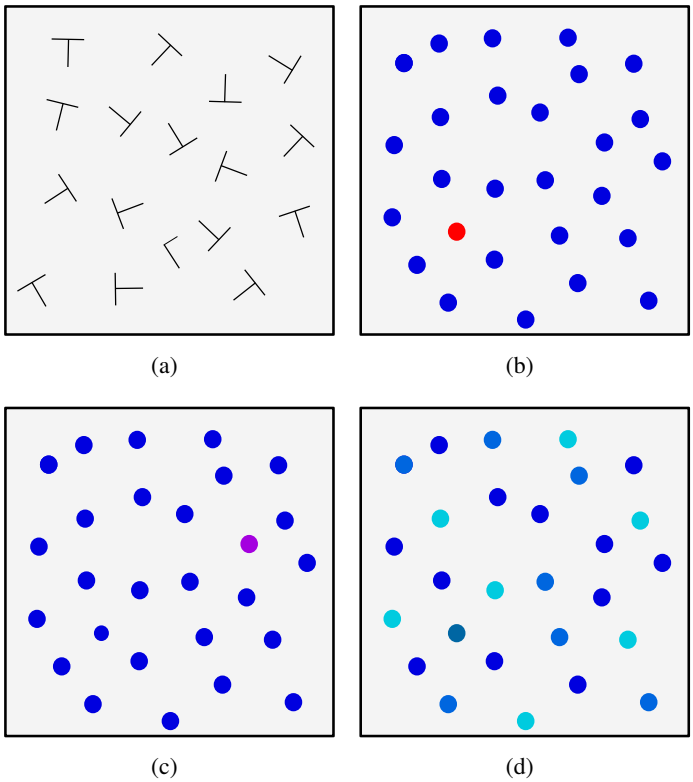


Figure 1.2: (a) Example of inefficient search (find the 'L' amongst the 'T's): no immediate pop out of the 'L', (b) Example of efficient search: pop out of the red circle, (c) Reduced pop out due to the similarity between the purple-blue target and the blue distractors, (d) Reduced pop out due to distractor-distractor dissimilarities.

against display size (or the RT$\times$Display Size function) is useful as a metric of search efficiency. Most instances of visual search, however, involve a combination of bottom up and top-down effects.

These effects do not occur in entirely separate processing stages but instead interact to determine performance. Top-down, knowledge-based processing can influence attentional guidance even when search is highly efficient [84]. Similarly, top-down knowledge of the relevant target feature can help observers to reduce or eliminate distraction from salient but irrelevant display items [39]. Top-down knowledge is applied through an attentional template [23], a form of visual working memory that makes up the visuospatial sketchpad [2], which can be used to filter out parts of the visual field when attending to a location, or as a map of features when searching for or attending to an object.

The phenomenon of a globally salient singleton object to pop out and the diminishing pop out observed with decreasing difference between targets and distractors is so accepted as be included in most modern computational models of visual search. Furthermore, the performance of each model at these tasks is a common metric for inter-model comparisons [21]. Feature Integration Theory [71] and Guided Search [85] support the idea of parallel feature processors updating a global saliency map [34, 35]. This map is a neural representation of the visual field with activations corresponding to a likelihood that a particular location contains an object of interest (i.e. a target) [87].

When parallel search fails to find the target, Guided Search uses the generated saliency map to inform the attentional deployment processes of likely next locations to search for the target, in descending order of activation strength [83]. Some form of Inhibition of Return (IOR) is needed to prevent the same location being attended to multiple times. One method for implementing IOR is given by Clarke *et al* [11], which uses two dimensional Gaussian masks to subdue activations in the saliency map at an attended location, with the strength of the mask decaying over time.

1.7 Feature Binding

The distributed representation of a scene in terms of the 'what' and the 'where' poses a problem of correctly binding together a perceived object and its position, particularly in a multi-object scene. For example, if a scene containing a square to the left of a circle is viewed, depictions of a square and a circle are determined by the ventral stream (AIT), while their positions are determined by the dorsal stream (Posterior Parietal (PP)). The binding problem arises as the distributed cortical representation allows the scene to be interpreted either correctly with the square to the left of the circle, or erroneously with the square to the right of the circle. Furthermore, a representation of objects as a collection of features in AIT presents a similar problem when resolving the properties of an individual object amongst many. Consider figure 1.3, in which complex shapes in AIT are represented by model neurons, with colour represented separately. The forward network causes the neurons indicating the presence of a square and a circle to activate, but also activates the neurons coding for red and green. The AIT representation is ambigu-

ous for this scene. However, the dorsal and ventral streams are connected in early visual areas, allowing selection of a feature to create spatially related activity in the dorsal stream. In turn, this spatial information from the dorsal stream on the locations of each feature allows the colour and form of each object to be bound together, such that the distributed representation for such a scene can be resolved. This process involves feature-based attention to select candidate locations in the retinotopic early visual areas, then spatial attention on these selected areas to inhibit distractor features. In the case of figure 1.3, attending to the square feature, will allow the colour at the same location as the square feature to remain, binding the two features together.

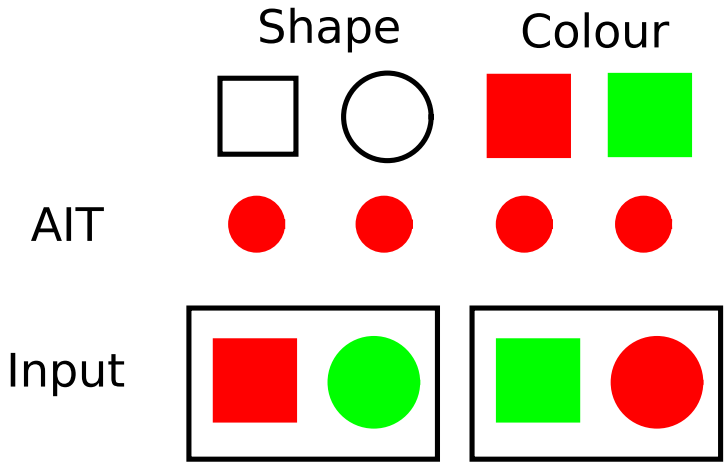


Figure 1.3: Example of an ambiguous representation of a visual scene in AIT. When presented with either of the inputs, a red square and green circle, or green square and red circle, the AIT representation is the same. This distributed representation requires binding to resolve.

1.8 Modelling Visual Attention

A number of studies have found that feature-based attentional effects occur across the visual field [79, 59, 60, 7, 52, 90]. The Sàenz *et al* studies [59, 60] used fMRI to test if feature-based attention produces global modulation of cortical populations. The studies presented overlapping fields of upwards or downwards moving dots in one visual hemifield to which the subject's attention was directed, and a single field of moving dots in the contralateral hemifield. Every twenty seconds subjects changed their attention between the upwards or downwards moving dots in the

attended to hemifield. This effectively alternated the trial conditions between attended and ignored stimuli moving in the same direction, or in opposite directions. Their results show that attention to a feature modulates responses of neurons sensitive to that feature across the visual array, including the ignored stimuli in the contralateral hemifield.

This thesis introduces a neural model of attention that postdicts this effect as neural activations generated by AIT neurons in a feedback network are applied throughout the visual field as a consequence of the inter-areal connection structure. The neurons in a reverse network coding for features are modulated by task-relevant information originating in higher areas, such as prefrontal cortex, through object and feature representations in AIT to the lower visual areas (V1). The model features widening neural receptive fields in subsequently higher visual areas, such that the receptive fields of neurons in AIT are the entire visual field. Furthermore, a recent study by Zhang and Luck [90], using a similar experimental paradigm to Sàenz *et al* but recording cortical activity via EEG, demonstrates this attentional modulation occurs only in cases where the attended to feature is in competition with distractors. We interpret this result as a consequence of binding: without neural activations from distractor features the representation in AIT is unambiguous, so the attentional neural cascade is not required.

1.9 Extant Models of Visual Attention

A number of models of visual attention exist. This section discusses two existing models with similar goals to this thesis.

1.9.1 Closed Loop Attention Model

The Closed Loop Attention Model (CLAM) of Visual Search proposed by van der Velde and de Kamps [78] includes the generation of saliency maps that are used to select a location to which attention is directed. CLAM consists of layers of neural networks that mimic the layered organization of the human visual cortex, including the two primary cortico-visual pathways identified by Mishkin *et al* [46]: the ventral pathway from area V1 (striate cortex) to the anterior inferotemporal

cortex (AIT) responsible for object recognition; and the dorsal pathway from area V1 to the Posterior Parietal (PP) region for spatial vision.

In CLAM, the binding problem of distributed features is resolved using a 'blackboard' [19] to tie together the distributed components of a scene using spatial information. The blackboard itself is simply the early layers of the visual streams, where the layout of the neurons map topographically to the retina (retinotopic), and so maintain good spatial congruity with the visual scene. As these areas are common to both the ventral and dorsal stream, interaction between them can occur [78]. The spatial map created in the ventral stream inhibits a spatial map of inputs to the dorsal stream in a retinotopic manner, allowing location information about a selected object to be passed into the dorsal stream in the form of a contrast map [77].

Visual search comprises a bidirectional flow of activation between the lower and higher visual areas, in terms of bottom up (stimulus driven) and top-down (attentional/user driven) processes. CLAM models these flows by two neural networks for each of the dorsal and ventral streams, with activations flowing from V1 to higher visual areas in one network, and an identical network propagating activations from higher areas to V1. Each pair of networks interact through local microcircuits via a disinhibition mechanism to enhance or suppress activations of features and positions [75]. In the bottom up flow, homogeneity in cells' receptive fields is penalized through layer local inhibition (e.g. the biased competition model of Desimone and Duncan [21]). Combined with larger receptive field sizes in higher visual areas, this mechanism can exhibit pop out of globally salient objects. A similar mechanism models the effect of attention via the top-down flow, except the top down maps generated by either the dorsal or ventral stream are primed for a cued location (or object/feature set). Guided Search and Feature Integration Theory [87] include an analogous processes of retrieving location information about objects of interest through interaction between feed-forward and feedback networks. Unlike CLAM, both Guided Search and Feature Integration Theory resolve the singleton object in the feed-forward pathway of the ventral stream through the biased competition model, although Feature Integration Theory models the interaction between the feed-forward and feedback networks as enhancing activations rather than suppression. The Normalization Model of Attention [54] uses a similar idea of competition in maps to determine where to direct attention, although the model has only a single stage, the authors posit that multiple stages of feature integra-

tion and attentional modulation in line with their model but occurring in each of the visual layers will fit their simulation data to the electrophysiological data from experiments.

CLAM is implemented using Multiple Instantiations of Interacting Neural Dynamics (MIIND) [16], an open source library of tools for creating neural network simulations. MIIND includes advanced features that can be used to convert a biologically unrealistic Artificial Neural Network (ANN) to more biologically plausible dynamical networks, with neural outputs being more accurately modelled with population firing rates [17, 40]. This conversion is necessary as the networks can only be trained as ANNs, but these networks do not correlate well with the properties of real cortical neurons [10].

1.9.2 Deco & Rolls' Model

Deco and Rolls presented a Gabor wavelet based neuro-dynamical model of cortical attention [20], which showed the shrinking of an AIT neuron's receptive field around an attended to object. The model was trained to recognize one of two objects in a cluttered visual environment. They presented the target object on a blank background, and overlaid on a natural scene, with a distractor object in either the same hemifield as the target, or the opposite hemifield. The model applied spatial and object-based attention to find one of the two objects.

The results showed the attentional modulation via a learned template reduced the effective receptive field size in AIT around the attended to object. This was a result of local competition among neurons coding for different form features, with a bias towards those neurons matching the attentional template, as predicted by the biased competition model [23].

The model presented in this thesis tried to replicate this behaviour using an architecture of cortical micro-circuits. However, the model presented by Deco and Rolls is only trained to recognize two different objects, while the model presented here uses four object categories, which increases the degree of mismatches between the stimulus and learned (attentional) representation. Their model also provides direct feedback to modulate the visual stimuli, whereas the model presented herein performs this modulation within cortical micro-circuits in intermediate layers of the ventral stream.

1.9.3 Dynamical Interacting Artificial Neural Network Application

The Dynamical Interacting Artificial Neural Network Application (DIANNA), was created to provide the flexibility missing from CLAM and MIIND for building complex interactions amongst layers of neural circuits. MIIND does not provide the ability to vary algorithm and circuit parameters within a single network. The large variation in connectivity of individual ANN nodes, or dynamical populations, does not allow a single set of algorithms or circuits to be tuned in a way that allows all neurons to remain sensitive to small changes in incoming activity. DIANNA allows increased control of these parameters by defining them over individual layers, where neural input is more uniform in scale.

In de Kamps and van der Velde [18] these limitations of MIIND were not reached, as each neural layer had the same dimension, and objects were only presented in one of four non-boundary locations, and the neural activity in each of these locations did not interfere with other locations (see figure 1 in [18]), to the extent that the network could have been split up into four separate networks for each quadrant. This thesis presents a model of translation invariance, so the neural activity caused by each input pattern suffers from interference from other inputs, as well as previous presentations of the same stimulus at different locations, which requires extensions to de Kamps and van der Velde's model. These extensions are not possible using the structural tools available MIIND, motivating the development of DIANNA.

A detailed presentation of DIANNA is given in chapter 2.

1.10 Thesis Outline

Chapter 1: Introduction (this chapter) Introduces cortical visual areas and visual attention modalities, and discusses models and theories of feature-based attention.

Chapter 2: DIANNA Describes the DIANNA software used to model the simulations and circuits presented in this thesis, and its motivations.

Chapter 3: Modelling Feature-based Attention with ANNs The architecture of the model is discussed at the level of interacting sub-networks and layers. Approaches to training the neural networks as top-down and bottom-up neural flows

are also discussed.

Chapter 4: Dynamical Modelling of Feature-Based Attention A review of the techniques applied in modelling the neural dynamics. Discussion of the neural dynamics and demonstration of the behaviour of Wilson-Cowan populations at steady states for a variety of squashing function algorithms. The developed model of visual attention is presented and evaluated against existing models of visual attention.

Chapter 5: Conclusion Closing remarks.

Chapter 2

Dynamical Interacting Artificial Neural Network Application

The cortex is composed of a number of functional regions (brain areas), each with a similar layered structure of cortical-columns, with recurrent connections between layers within cortical columns, and between functional regions. A software application was created for this PhD which allows modelling this regional and micro-circuit structure with neuro-dynamical populations, and allows these networks to be trained as Artificial Neural Networks (ANNs), prior to conversion to dynamical networks. The software is called Dynamical Interacting Artificial Neural Network Application (DIANNA).

2.1 Motivation for DIANNA

The general approach used by DIANNA is to create layers of dynamical circuits and specify the connections between layers. All circuits within a layer are identical, but each circuit member population can be parametrized individually. The simplest approach of modelling the structure of the cortex with DIANNA, is to model functional regions as layers, and cortical columns as circuits. However, it is also possible to model brain areas using a number of layers, and this approach is useful when arbitrary connections are desired between circuits and layers. This multi-layered approach was used to generate the networks in chapter 4, as the forward and reverse dynamical networks were created from trained ANNs, while the

intermediate (disinhibition) layer was implemented directly as a layer of dynamical circuits.

DIANNA allows the creation and training of ANNs, and their subsequent conversion to dynamical networks, maintaining the layer structure of the ANNs, but replacing each ANN node with a dynamical circuit. Once a layer and connection structure has been established, DIANNA can train the ANN using a variety of techniques, including backpropagation, and gradient descent. Once trained, the ANNs can be converted to dynamical networks by replacing each ANN node with a circuit, with a separate circuit design for each ANN. The conversion copies the learned weights between ANN nodes to the appropriate circuit members via declarative or programmatic rules. The conversion of ANNs to dynamical networks follows the method described by de Kamps *et al* in [17], but allows the mapping of positive and negative weights to be performed between arbitrary circuit designs by defining circuit populations with input roles which accept either positive or negative weights.

Once a dynamical network has been constructed, multiple simulations may be run against this network by varying the network's inputs. These inputs can be obtained directly from images, read from Extensible Markup Language (XML) files, or hand coded. A completed simulation can be visualized using DIANNA to show the interactions over time of the network, from network, layer or circuit based perspectives.

DIANNA is based on the MIIND software [16], but is focused on the creation of layered dynamical neural systems of arbitrary design. MIIND is wider in scope (i.e. MIIND now provides simulating large populations of neurons via the population density equation described in [15]), and as such does not offer the same flexibility as DIANNA in parametrization of connectivity and activation functions. Running simulations in DIANNA is also highly parallel, scaling to use the number of available Central Processing Units (CPUs). Also, training ANNs and simulations of dynamical networks in DIANNA are highly stochastic, with the the order of updates being randomized for each network step, using a time-based seed to the Random Number Generator (RNG) for each network creation, training and simulation run. While this prevents the exact duplication of results between runs, it also prevents results relying on a given initial environment.

The backpropagation training algorithm in DIANNA is also more advanced than

that of MIIND, allowing the use of thresholds (biases) for ANN nodes, batch and online training, momentum, and reducing the learning rate over the course of training. There are also additional activation functions available in DIANNA over the sigmoid function implemented in MIIND, including tanh and a Gaussian sigmoid function. Where these activation functions range from -1 to 1, half-range functions from 0 to 1 are implemented for use with dynamical systems. The dynamical systems are based on Wilson-Cowan dynamics [81, 82] which model the spike rate of populations, so negative values would suggest negative spike rates, which are implausible.

2.2 Architectural Overview

A DIANNA instance is organized through a management object: The `Project`. This section describes the architecture of the `Project` as interacting high level objects.

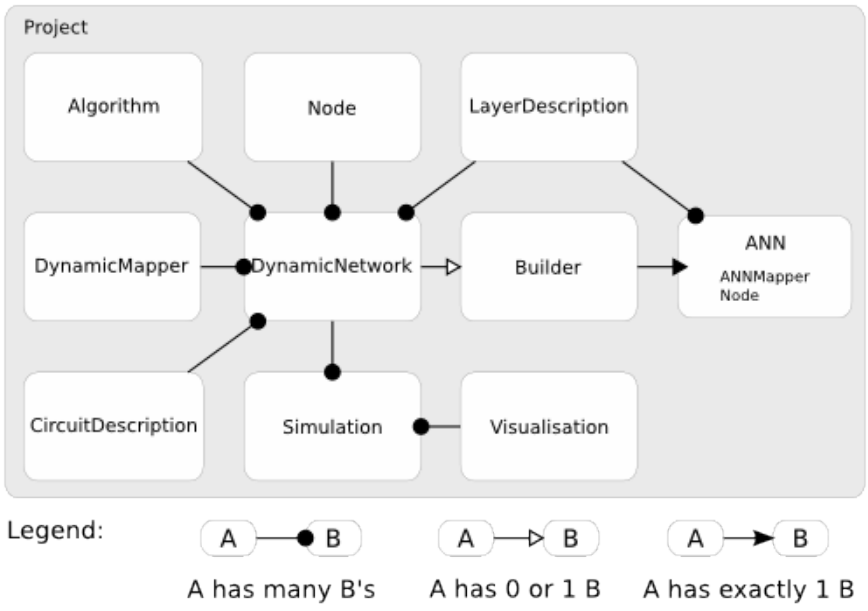


Figure 2.1: High level view of the relationships between a `Project` object and other classes.

2.2.1 The `Project` object

The `Project` object is the container for a project and all of its artifacts, such as log files, networks and simulations. All paths in a `Project` are implicitly relative to the `Project` file, managed by this `Project` object.

In addition to managing a `Project`'s artifacts, the `Project` object acts as a registry for all objects created within a `Project`. This allows all interacting objects to be managed by name. Internally the `Project` object manages separate look-up tables for:

- LayerDescription objects
- ANN Network builder objects
- CircuitDescription objects
- Algorithm objects
- Dynamic networks
- Dynamic mappers
- Simulation descriptions
- Visualization descriptions

When instances of these classes are created they can be added to the `Project` with the `addX` method on `Project`, where *X* is the object type to be added. These methods add the instance to the appropriate registry under the object's *name*. Future references to these objects can then be retrieved from the `Project` by name. The :ref:examples_page page shows many examples of this approach.

The `Project` manages a single collection of `Node` objects shared for all `DynamicNetwork` objects in the `Project`. The number and type of nodes is determined by the `LayerDescription` objects referenced by each `DynamicNetwork`. A `DynamicNetwork` maintains `AbstractAlgorithm` objects for excitatory and inhibitory connections, and a `CircuitDescription`. This design means that all layers in a `DynamicNetwork` have the same dynamics and circuit structure. However, a `Project` does not limit the number of `DynamicNetwork` objects, so greater flexibility can be achieved by having a `DynamicNetwork` for each layer, with algorithms specified as desired.

2.2.2 `DynamicNetwork` Structure

A `DynamicNetwork`'s structure is defined by `LayerDescription` and `CircuitDescription` objects. A `LayerDescription` specifies the number of circuits in a two-dimensional sheet, as height and width, and has a third dimension specifying the number of features in this layer. When the `DynamicNetwork` is constructed, the `Project` allocates `Node` objects calculated as the number of features multiplied by the number of circuits in each feature, multiplied by the number of nodes in each circuit.

`CircuitDescription` objects consist of `CircuitNodeRole` objects which describe what role a `Node` plays within the circuit. Roles with the `input` property allow incoming connections to the circuit, while roles with the `output` property are used for visualization of the circuit's activity. The `CircuitDescription` includes connectivity information for all its internal connections.

After construction of a `DynamicNetwork` each layer has been converted to circuits. However, with the exception of intra-circuit connections specified in the `CircuitDescription`, the network is disconnected. To create inter-circuit and inter-layer connections `DynamicMapper` objects are needed. These objects specify how two layers are linked together through their circuits. Once a `DynamicNetwork` has been converted to `Node`s, `DynamicMapper`s can be used to create arbitrary connections between layers of different networks.

2.2.3 Applying Patterns to a `DynamicNetwork`

A `LayerDescription` specifies whether a layer is an input or output layer. The `CircuitDescription` also specifies particular nodes as input or output nodes for that circuit. Where a node is both an input node and exists in an input layer, input patterns can be applied to the `DynamicNetwork` during a simulation.

See section 2.3.2 for details on the training file format.

2.2.4 Training Neural Networks

A `Project` manages `Simulations` of `DynamicNetwork`s by evolving each node by a small time value. A small time period improves accuracy of the simulation

while increasing computational overhead. This overhead makes it infeasible to train dynamical networks in the same way as ANNs. However, the alternative to training an ANN is to specify all of the weights by hand, which again, is infeasible for larger networks.

However, if the network is partially ordered, evolving this way is unnecessary as the inputs to a node will not change during the time-course of a simulation. The Backpropagation algorithm for training ANNs requires this partial ordering property for the correct allocation of 'blame' to the source of the errors. So if even part of a network is partially ordered it would be useful to be able to train this part before weaving it into the larger population of networks managed by the `Project`. A `DynamicNetwork` has a `AbstractNetworkBuilder` parameter for defining an ANN from which connection weights can be extracted.

2.2.4.1 Training Artificial Neural Networks

The Backpropagation algorithm requires parameters not specified for `DynamicNetwork` objects which do not have a backing ANN. These parameters are used to configure a `AbstractNetworkBuilder` instance. The code below shows how to configure a network trained via Backpropagation:

```
ann_param = SigmoidParameters(rate_max=2.0, noise=0.001)
ann_alg = GaussianAlgorithm("ann", ann_param)
project.addAlgorithm(ann_alg)
ann_mappers.append(BubbleANNMapper(0, 0, 1, 0, 1.0, 2.))

params = TrainingParameters(learning_rate = 1.0,
            momentum = 0.0,
            max_epochs = 25,training_set_mse = 1e-3,
            log_file_name = fwd_log_file_name,
            log_file_resolution = 5,
            training_set_file_name = training_set_file_name,
            use_batch = True, use_bias = False)

fwd_builder = BackpropNetworkBuilder(project,
            "fnet",
            trained_fwd_net_file_name,
```

```
            params,
            ("V1", "V2", "V4", "PIT", "AIT"),
            ann_mappers,
            ann_alg)

fwd_net_description = DynamicNetworkDescription("fnet",
            "perceptron",
            ("V1", "V2", "V4", "PIT", "AIT"),
            "exc", "inh",
            fwd_builder)
project.addDynamicNetworkDescription(fwd_net_description)
```

See `TrainingParameters` for details of accepted parameters.

A `TrainingParameters` object is populated and used to configure a `BackpropNetworkBuilder` object. This object will create a network named "fnet" with the previously defined layers named "V1", "V2", "V4", "PIT", and "AIT". The layers are connected via the mappers defined in `ANNMapper`, only one of which is shown in the code snippet. A `DynamicNetwork` is then created with a named `CircuitDescription` "perceptron" and the `AbstractNetwork-Builder` object, passed as the final constructor parameter.

After adding the `DynamicNetwork` to the `Project` the network not yet is available for use. At this time the `NeuralNetwork` has not been constructed, so no `Node`s have been created. This is because the `Project` lazy loads `Neural-Network` s as required. In order to create this `NeuralNetwork` simply call:

```
Project.getNetwork("fnet")
```

This will return the in-memory `NeuralNetwork` if it exists, or load the `NeuralNetwork` from disk, if it exists, or build the `NeuralNetwork` from its description otherwise.

At this stage the `NeuralNetwork` is constructed, but its weights will be random. The `NeuralNetwork` is trained via its `AbstractNetworkBuilder` object:

```
network.builder.trainNetwork()
```

This could take some time, depending on many factors, such as the size of the network, the error rate sought, the number of training patterns, among many others.

During training, progress information is written to a log file at the resolution specified in the `TrainingParameters` given to the The `AbstractNetworkBuilder` instance used to construct the `NeuralNetwork`.

Once trained, the `NeuralNetwork` is written to disk at the location specified in the `AbstractNetworkBuilder` constructor. Subsequent fetches of this network will try to load this previously trained `NeuralNetwork`, and if not found will create and train a new `NeuralNetwork`.

An ANN cannot be used directly with `DynamicNetwork`s, and is transparently converted into a `DynamicNetwork` of circuits as specified by the `CircuitDescription` name given to the `DynamicNetworkDescription` constructor. Internally this is done by creating `ANNToDynamicMapper` objects which map ANN nodes to circuit roles in the new `DynamicNetwork`. ANNs can output positive and negative values. When mapped to a `DynamicNetwork` which outputs spike rates, negative values are nonsensical. During the conversion the `ANNToDynamicMapper` maps positive output activity of the ANN node to a positive input node in the circuit, and negative output activity to a different circuit node. `CircuitNodeRole`s also have the `positive` property, which signifies whether the output should be considered as positive or negative in sign, despite the output always indicating a positive spike rate.

2.3 Network Inputs

The network receives input through specialized input layers. These layers contain neurons which output a fixed spike rate which is determined by a variety of methods: direct input, XML-based input or extracted from images via input filters (Gabor filters, and colour filters).

2.3.1 Direct Input

Ad-hoc input patterns can be created through visualizing artificial neural networks. By selecting individual nodes in input layers, an output rate can be specified directly. This method allows arbitrary patterns to be created and immediate feedback of the network's response to be visualized. At any point a dynamical simulation can be created from patterns entered using this method and visualized separately

for investigation of more complex neural dynamics. The direct input method also allows editing of input patterns specified by alternative methods.

2.3.2 XML-based Input

For the trained network to generalize on novel inputs it must be trained on a large number of inputs and their matching output patterns. Matching pairs of input and output patterns for each neural network can be specified in an XML format and passed to the network during training, or specified as input patterns for a dynamical simulation. Collections of these input-output pairs can be created for training, generalization and validation patterns. These patterns can be marshaled to XML for storage in pattern files for reuse in future training or simulations.

The format of the XML is simple and the schema is provided below. Every input neuron has its activity specified directly in the XML. The simplicity of this approach allows pattern files to be reused for multiple networks and simulations rather than be tied to network for which is was originally created.

The training file is a simple XML format, allowing hand editing of training data or generation by external tools.

The training file has a root node `training_set` with three child nodes of:

- `training_units`: Patterns applied during training for the network to learn from.
- `generalization_units`: Applied after each training epoch to test how well the network can generalize to untrained input.
- `validation_units`: Applied after training is complete to test how well the network performs on completely unseen data.

The `generalization_units` and `validation_units` can both be empty tags.

Each training unit consists of two patterns, the first is the input pattern, the second the output pattern. The patterns must have the same dimensions (`width`, `height`, and `num_features`) as the layers they will be applied to.

The `DATA` portion of the pattern is space separated input rates for each node. The data is mapped to the input layer in row order, feature by feature. The following

pseudo-XML describes the structure at the `training_set` level:

```
<training_set>
      <training_units>
            ...
          <training_unit />
          ...
      </training_units>
      <generalisation_units>
            ...
          <training_unit />
          ...
      </generalisation_units>
      <validation_units>
            ...
          <training_unit />
          ...
    </validation_units>
</training_set>
```

The `training_units`, `generalisation_units` and `validation_units` tags contain `training_unit` tags with the following format:

```
<training_unit>
  <pattern height="1" width="1" num\_features="5">DATA</pattern>
  <pattern height="1" width="1" num\_features="5">DATA</pattern>
</training_unit>
```

For example, given the `LayerDescription`:

```
<layer_description name="layer" height="2" width="2"
      num_features="3"/>
```

This produces 12 nodes:

```
A B   E F   I J

C D   G H   K L
```

Given the pattern:

```
<pattern height="2" width="2" num_features="3">
   0 1 2 3 4 5 6 7 8 9 10 11
</pattern>
```

The inputs would be applied:

```
A B   E F   I J
0 1   4 5   8 9

C D   G H   K L
2 3   6 7  10 11
```

2.3.3 Image-based Input

2.3.3.1 Gabor filters

Input patterns can also be applied to the network as colour images (Red, Green, and Blue channels), and processed via 2-dimensional Gabor filters into appropriate input for each feature type [13, 14]. Gabor filters are comprised of a sinusoidal wave with a given direction and wavelength, and a Gaussian mask with defined radius and standard deviation. This mask can be applied such that a only a single half-cycle of the wave is unmasked, thereby allowing the Gabor filter to only pass lines of a single orientation (and multiple thicknesses, if multiple wavelengths are used to define the sinusoidal waveform). The waveform includes real and imaginary components, which are slightly offset from each other, such that in combination, the Gabor filter is capable of differentiating between lines which pass from dark to light (and *vice versa*). This allows a sense of which edge of the detected line is internal or external to a solid object to be imparted to the network, assuming that the background remains constant across the image. Figure 2.2 shows example output of 2-dimensional Gabor filters sensitive to lines of 0°, 45°, 90° and 135°, applied to an image containing the outline of a square.

Each feature within an input layer can have Gabor filters attached to process the images for features of interest. Gabor filters can be constructed to detect lines of a given orientation and spatial frequency and input to the receptive field of each input neuron. For each form feature, Gabor filters with a common orientation but

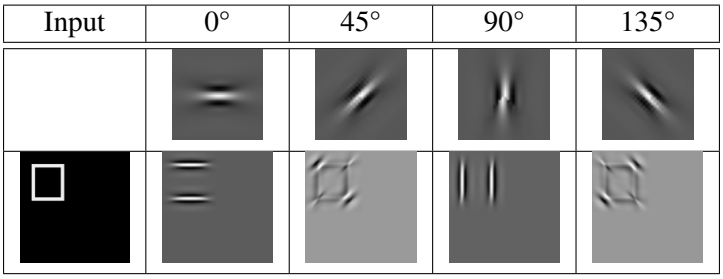


Figure 2.2: An input image of the outline of a white square applied to Gabor filters with orientations of 0°, 45°, 90°, and 135° (middle row), and their outputs (bottom row).

different spatial frequencies can be constructed and applied via a `FilterBank` object, which consolidates all contained Gabor filters into a composite which applies input in the same way as a single Gabor filter, but provides detection of different widths of line to the same input neurons. The filter bank may be comprised of log-Gabor filters [25], where the spatial frequencies of each filter are separated on a logarithmic scale, which perform better on natural images and are more robust to multiple spatial frequencies than linear Gabor filters [25]. The idea of multiple Gabor filters with different spatial frequencies and orientations but centred on the same spatial location is termed a Gabor Jet and has been proposed as the basis for V1 hypercolumns [9].

The receptive field properties of each input neuron such as its size, shape and overlap can be set for the input layer independently of the Gabor filter. The three colour channels are collapsed to a grey-scale image for use with Gabor filters, as the colour components are not generally meaningful in the context of line orientations. However, images of each colour channel may be applied separately, if combinations of colour and form are needed.

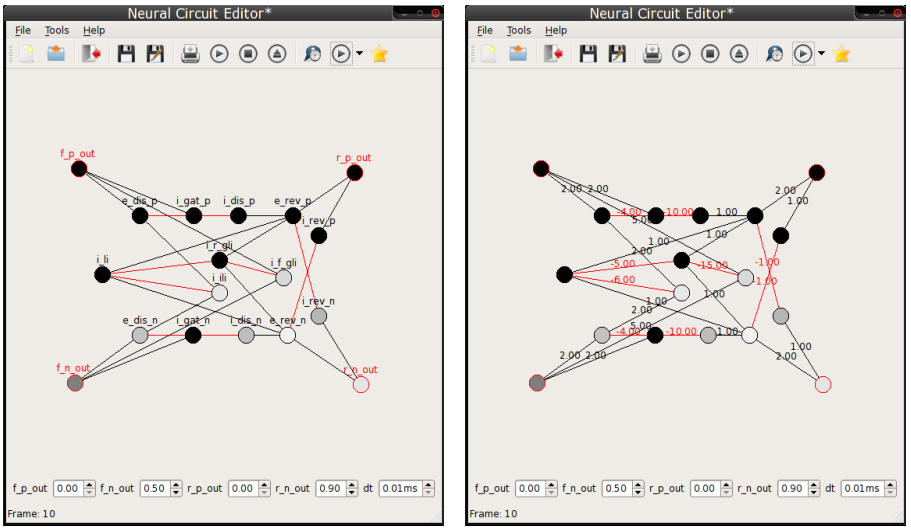
By using Gabor filters, the suppression of neural activity to anti-preferred stimuli, as discussed in section 1.4, can be performed by the application of Gabor filters sensitive to orthogonally oriented lines as negative inputs to the same form feature.

2.3.4 Colour filters

In order to model the colour modality of feature based attention ‘Colour filters’ allow the extraction of single colour channels from images to be applied to input

layers. The receptive field properties of each input neuron can be set independently, as described previously for Gabor filters (see 2.3.3.1).

2.4 Circuit Editor

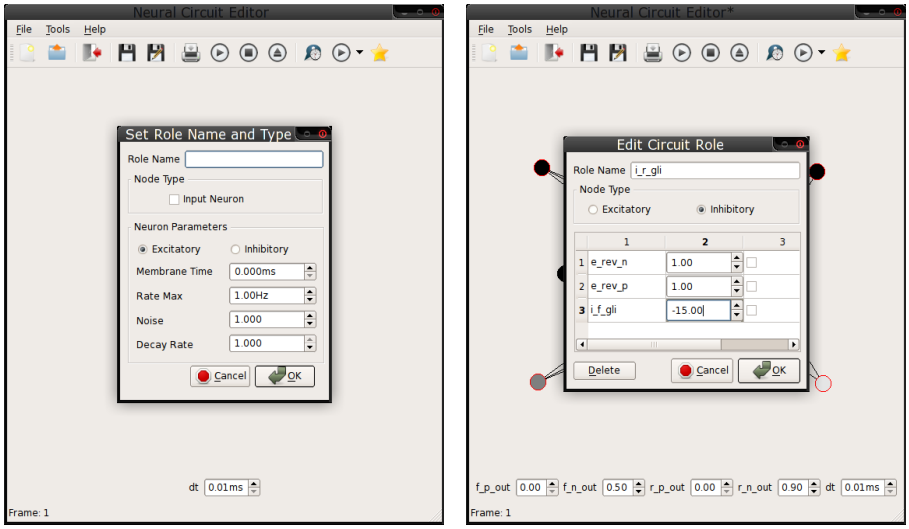


(a) (b)

Figure 2.3: Circuit editor application main interface, showing the disinhibition with lateral inhibition circuit: a) names of circuit members, b) connection weights. Red connections/weights are inhibitory. Red outlined nodes are input nodes to test the circuit and are not part of the circuit itself.

In order to develop and test the functionality of cortical circuits, DIANNA includes a graphical application for the creation and evaluation of circuits of Wilson-Cowan populations (figures 2.3 and 2.4).

The circuit editor application allows arbitrary connection structures to be designed and inputs applied to check the circuit's behaviour. Inputs and weights can all be varied independently. The designed circuits can then be output as XML for inclusion into a DIANNA project file, or printed as Portable Network Graphics (PNG) bitmaps or vector graphics for publication. The images of circuits shown in this thesis were all generated using this tool.



(a) (b)

Figure 2.4: Circuit editor application: a) Creating a new circuit member b) Editing an existing circuit member and its connection weights.

2.5 An example DIANNA project

This section provides an annotated example of using DIANNA to train an ANN of 3 layers to solve the Exclusive OR (XOR) logical operation. This ANN is then converted to a dynamical simulation.

First a `Project` object is created to hold all of the components of the ANN and simulations:

```
project_desc = ProjectDescription("xor_network")
project_desc.setDescription(
  "XOR network to demonstrate algorithm's convergence speed")
project = Project(project_desc)
```

Three layers are then added to the `Project`. The first two layers (named "First" and "Mid") are 2 neurons high and 1 neuron wide. The "First" layer also acts as the input layer. In DIANNA terms, this means inputs are applied directly to layer "First", rather than via a dedicated input layer, as is commonly seen in neural network examples. Therefore the weights between layers "First" and "Mid" will be altered by training, despite layer "First" also being an input layer.

```
project.addLayerDescription(
   LayerDescription("First", 2, 1, 1, is_input=True)
   )
project.addLayerDescription(
   LayerDescription("Mid", 2, 1, 1)
   )
project.addLayerDescription(
   LayerDescription("Last", 1, 1, 1)
)
```

Next, some data for training is created. This is done algorithmically, and creates 4 input/output pattern pairs, matching the truth table for the XOR logical operator. The training patterns are added to a `TrainingSet`, which is then wrapped in a `TrainingSetInputFilter` object, which specifies to which layers the input and output patterns correspond.

```
tus = [] for x in range(2):
```

```
      for y in range(2):
         inPat = Pattern(2, 1, 1)
         inPat.setAt(0,0,0,x*0.9)
         inPat.setAt(0,1,0,y*0.9)
         result = 0.0
         if x != y:
            result = 0.9

         outPat = Pattern(1, 1, 1)
         outPat.setAt(0,0,0,result)
         tus.append(TrainingUnit(inPat, outPat))

## Wrap the TrainingUnits in an InputFilter
training_set = TrainingSet(tus, [], [])
xor_input_filter = TrainingSetInputFilter(
   project,
   "First", "Last",
   training_set)
```

A `ZeroBasedSigmoidAlgorithm` is then defined for all of the ANN nodes. By varying the algorithm and parameters, the convergence speed can be greatly improved. Training parameters are then specified. In this example, the training will be completed when the mean-squared error of the training patterns is less than 1×10^{-3}. The bias, or threshold, of the ANN nodes is not trained. Momentum is not used, as the patterns are applied randomly, and on-line training is used, whereby the network weights are updated after each training pattern has been propagated through the network (specified by `use_batch=False`).

The algorithms and `TrainingParameter`s are used to create a `BackpropNetwork-Builder`. This object will create the structure of the network, and will apply the backpropagation algorithm during training.

```
## Default parameters: noise=1, power=1
ann_param = AlgorithmParameters()
## Converges in ~1480 epochs
ann_alg = ZeroBasedSigmoidAlgorithm(
            "zero_based_sigmoid_n1_p1", ann_param)
```

```
project.addAlgorithm(ann_alg)

## Train forward net by setting up common training parameters
params = TrainingParameters(
            learning_rate = 1.0,
            momentum = 0.0,
            max_epochs = 10000,training_set_mse = 1e-3,
            input_filter = xor_input_filter,
            log_file_name = "log.txt",
            log_file_resolution = 1,
            use_batch = False, use_bias=True)
## Training params are used by a network builder object
xor_builder = BackpropNetworkBuilder(project,
         "xor",
         ann_filename,
         params,
         ("First", "Mid", "Last"),
         [], # Empty mappers object, as weights create manually
         (ann_alg.getName(), # List of algorithms for each layer
         ann_alg.getName(),
         ann_alg.getName()) )
## Create the nodes
xor_builder.build()
```

The network is then constructed. However, at this point each node in the network is disconnected. DIANNA is more suited to creating layered network architectures with a common connection structure between nodes in each pair of layers. This is performed via the creation of `ANNMapper` objects, which are passed to the `BackpropNetworkBuilder` during its construction. As the XOR network is small, the initial weights are specified by hand.

```
## Get the network and nodes
net = xor_builder.getNetwork()
node_a = net._nodes[0]
node_b = net._nodes[1]
node_p = net._nodes[2]
node_o = net._nodes[3]
```

```
node_z = net._nodes[4]

## Manually create the connections as (predecessor_node, weight)
node_p.addIncoming(node_a, 2.0)
node_p.addIncoming(node_b, 3.0)
node_o.addIncoming(node_a, 1.0)
node_o.addIncoming(node_b, 2.0)
node_z.addIncoming(node_p, 2.0)
node_z.addIncoming(node_o, -1.0)
```

An equivalent structure, but with random weights between the nodes of each adjacent layer, could be created with the following `ANNMapper` objects:

```
first_to_mid_mapper = AllToAllRandomANNMapper(
   "First", 0,
   "Mid", 0,
   3)
mid_to_last_mapper = AllToAllRandomANNMapper(
   "Mid", 0,
   "Last", 0,
   3)
```

In this example, `first_to_mid_mapper` will create connections between all nodes in the "First" layer, to all nodes in the "Mid" layer, with a random weight in the range -3.0 to 3.0. Likewise, `mid_to_last_mapper` will create random weights between the middle and output layers within the same range.

Once the connection structure has been specified, the network is trained.

```
xor_builder.trainNetwork()
```

At this point, the ANN has been created, and the weights have been trained for the network to perform the XOR operation on its inputs. The code following this point of the example program demonstrates creating dynamical networks and simulations with DIANNA.

A Perceptron circuit is created using separate `PositiveZeroBasedSigmoidAlgorithms` for the excitatory and inhibitory populations of the circuit. These algorithms are created with identical parameters, except for the membrane time

constant (the `time_membrane` parameter). A plot of these sigmoid functions is show in figure 4.1.

```
exc_param = AlgorithmParameters(
                time_membrane=1e-2,
                rate_max=100.0,
                power=1.0, noise=1.0,
                max_update_step=5e-4)
exc_sigmoid_alg = PositiveZeroBasedSigmoidAlgorithm(
               "exc_sigmoid_alg", exc_param)

inh_param = AlgorithmParameters(
                time_membrane=5e-3,
                rate_max=100.0,
                power=1.0,
                noise=1.0,
                max_update_step=5e-4)
inh_sigmoid_alg = PositiveZeroBasedSigmoidAlgorithm(
                "inh_sigmoid_alg", inh_param)

## Add them to the project
project.addAlgorithm(exc_sigmoid_alg)
project.addAlgorithm(inh_sigmoid_alg)
```

The Perceptron circuit is then defined, specifying these algorithms for each population in the circuit. These populations are created via `CircuitNodeRole` objects, as each population plays a different role within the circuit. The Perceptron circuit, and the roles each population plays within it are described in detail in section 4.2.1.

```
perceptron_sigmoid = CircuitDescription(
name="perceptron", number_of_nodes=6)
e_p = CircuitNodeRole("e_p", # Name
                 x_pos=3.0, y_pos=-2.0, z_pos=0.0, # Position
                 algorithm_name="exc_sigmoid_alg", # Algorithm
                 is_excitatory=True, # Inhibitory/Excitatory
                 is_input=True) # Receives input to the circuit
i_p = CircuitNodeRole("i_p", x_pos=-3.0, y_pos=-2.0, z_pos=0.0,
                 algorithm_name="inh_sigmoid_alg",
```

```
                is_excitatory=False, is_input=True)
e_n = CircuitNodeRole("e_n", x_pos=1.0, y_pos=-2.0, z_pos=0.0,
                algorithm_name="exc_sigmoid_alg",
                is_excitatory=True, is_input=True,
                is_positive=False)
i_n = CircuitNodeRole("i_n", x_pos=-1.0, y_pos=-2.0, z_pos=0.0,
                algorithm_name="inh_sigmoid_alg",
                is_excitatory=False, is_input=True,
                is_positive=False)
p_out = CircuitNodeRole("p_out", x_pos=3.0, y_pos=-2.0, z_pos=0.0,
                algorithm_name="exc_sigmoid_alg",
                is_excitatory=True, is_output=True)
n_out = CircuitNodeRole("n_out", x_pos=-3.0, y_pos=-2.0, z_pos=0.0,
                algorithm_name="exc_sigmoid_alg",
                is_excitatory=True, is_output=True,
                is_positive=False)

## Set up intra-circuit connections
p_out.addIncoming("e_p", 2.0)
p_out.addIncoming("i_n", -2.0)
n_out.addIncoming("e_n", 2.0)
n_out.addIncoming("i_p", -2.0)

## Add roles to circuit description
perceptron_sigmoid.addCircuitNodeRole(e_p)
perceptron_sigmoid.addCircuitNodeRole(i_p)
perceptron_sigmoid.addCircuitNodeRole(e_n)
perceptron_sigmoid.addCircuitNodeRole(i_n)
perceptron_sigmoid.addCircuitNodeRole(p_out)
perceptron_sigmoid.addCircuitNodeRole(n_out)

## Register circuit description in the project
project.addCircuitDescription(perceptron_sigmoid)
```

Next we create a `DynamicNetwork`. The network consists of the previously defined layers, “First”, “Mid” and “Last”, and the circuits used to replace each ANN

node in these layers. The `DynamicNetwork` constructor is passed the `Builder` object used to create the earlier ANN, which is used by the `DynamicNetwork` object to build its internal network structure, copying the weights from the trained ANN. The `DynamicNetwork` is created at first use, so after the dynamic network is defined, we *get* it from the `Project` object, which causes the network to be constructed

```
## Define a dynamic network.
dnet_description = DynamicNetworkDescription(
       "xor", # Name
       ("perceptron", "perceptron", "perceptron",),
       ("First", "Mid", "Last"),
       xor_builder,
       dynamic_inputs=False)
project.addDynamicNetworkDescription(dnet_description)

## Create the dynamic network
project.getDynamicNetworkByName("xor")
```

A simulation consists of constructed `DynamicNetwork`s and `InputPattern`s to apply to those networks. An `InputPattern` requires parameters for a `Pattern` object, the names of the network and layer to which the `Pattern` is to be applied, and the time at which to apply it. A list of `InputPattern`s to use in a simulation is provided to the `SimulationParameter` constructor. Further parameters to the `SimulationParameter` constructor are shown in the code snippet below.

The simulation is run from the `start_time` to the `end_time` in time steps of `network_step_time`. The networks are updated at `update_time` intervals, and the state of the networks is written to a results file at `report_time` intervals. Progress information is written to a log file at `log_time` intervals. In the example below, the simulation runs for 100ms, and the state of the simulation is recorded every 1ms, resulting in 100 snapshots of the state of the networks being recorded.

```
## Set the input to the simulation to be the second pattern
input_patterns = []
input_patterns.append(InputPattern(
                 "xor", "First", 0.0,
                 xor_input_filter.getInputPattern(1)))
```

```
## Define a simulation
simulation_description = SimulationParameter(
             name="sim",
             max_iterations = 100000,
             start_time=0.0,
             end_time=0.1,
             report_time=1e-3,
             update_time=1e-4,
             network_step_time=1e-5,
             log_time=1e-2,
             input_patterns=input_patterns,
             progress_file_name=simulation_progress_filename,
             output_file_name=simulation_filename)

## Add the simulation description to the project
project.addSimulationDescription(simulation_description)
project.createConnections()

## Run all simulations
project.runSimulations()
```

In order to view the recorded simulation data, a `Visualisation` is created containing each layer to be visualized, and a 2D position of that layer. The `VisualisationParameters` object sets colours used for the visualization, and the initial camera position and zoom-level.

```
## A sample visualisation
vis_params = VisualisationParameters(
                  (1.0, 0.0, -1.0),
                  (0xFF0000, 0xFFFFFF, 0x0000FF, 0xF0F0F0),
                 SpatialPosition(), 5.0)
first_vis_layer = SpatialLayerDescription(project,
                'xor', 'First', SpatialPosition(0, 0))
second_vis_layer = SpatialLayerDescription(project,
                'xor', 'Mid', SpatialPosition(75, 0))
last_vis_layer = SpatialLayerDescription(project,
                'xor', 'Last', SpatialPosition(150, 5))
```

```
vis1 = Visualisation(project, 'vis1', "sim", vis_params,
          (first_vis_layer, second_vis_layer, last_vis_layer ))
project.addVisualisationDescription(vis1)
```

Images of the simulation at frame 100, and plots of the activity of the output node are shown in figure 2.6 and 2.7. Figure 2.5 shows the trained ANN with each training pattern propagated through the network.

For larger networks and simulations, the creation of connections can be very time-consuming. This connection structure can be written to a cache and used for subsequent simulations. If the cache file exists, the connections will be read from it, rather than created at runtime.

```
## Write out the connection structure to a cache file
project.writeDynamicNetworksToCacheFile()
```

Finally, the `Project` object can be written to disk as an XML file.

```
## Write the project to disk using a ProjectWriter object
writer = ProjectWriter(project)
writer.writeProject(project_filename)
```

2.5.1 Project XML Format

The project XML file is the central configuration location for the declarative definition of a DIANNA project, simulations and visualizations. An example of the XML file format is given below, which is equivalent to the coded example described in section 2.5. The XML may be hand coded and provided to DIANNA's `Project-Runner` application, along with the name of a simulation to run, and the number of CPU cores to use for the simulation. The XML may also be created from the programmatic approach by passing the `Project` object to a `ProjectWriter` as shown in the previous section.

```
<?xml version='1.0' encoding='UTF-8'?>
<project name="xor_network">
 <project_description>
 XOR network simulation with Perceptron Circuits
 </project_description>
```

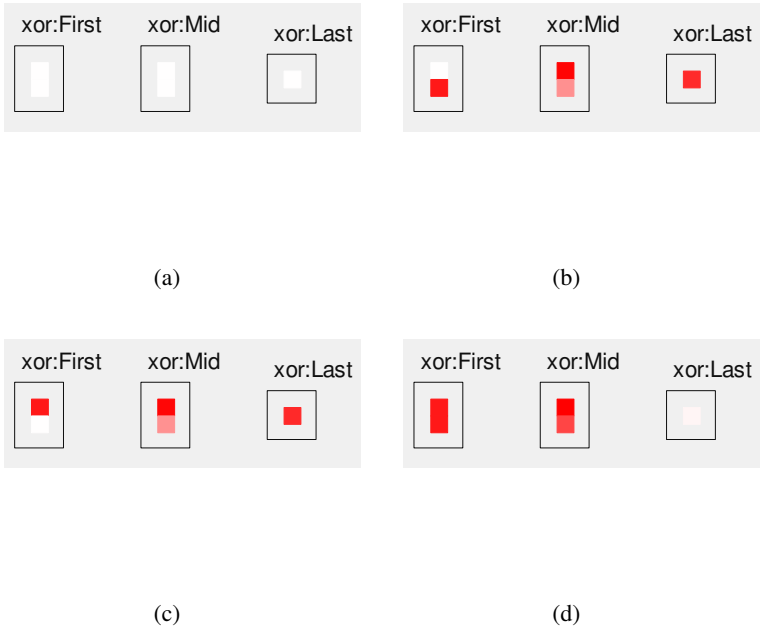


Figure 2.5: ANN images for the XOR-Network. Input to the network is applied to the two left hand nodes, and output is the right hand node. (a) Input of 0,0 shows 0 output. (b) Input of 0, 1 shows positive output. (c) Input of 1, 0 shows positive output. (d) Input of 1, 1 shows 0 output. Red shows strong positive output, white 0 output. Negative output is set to blue in the `Visualisation` object, but is absent in these results.

```
<layers>
  <layer_description name="First"
        height="2" width="1"
        num_features="1" input="True"/>
  <layer_description name="Mid"
        height="2" width="1" num_features="1"
        input="False"/>
  <layer_description name="Last"
        height="1" width="1"
        num_features="1" input="False"/>
</layers>
<algorithms>
  <zero_based_sigmoid_algorithm name="zero_based_sigmoid_n1_p1">
    <algorithm_parameter rate_max="1.0" noise="1.0"
```

Figure 2.6: Simulation images for the XOR-Network at the end of the simulation (100ms, frame 100). Input to the network is applied to the two left hand nodes, and output is the right hand node. (a) Input of 0Hz to both input population of layer "First". (b) Input of 0Hz to the top and 90Hz to the bottom input populations of layer "First". (c) Input of 90Hz to the top and 0Hz to the bottom input populations of layer "First". (d) Input of 90Hz to both input layer populations. Red shows strong positive output, white 0 output. Negative output is set to blue in the `Visualisation` object, but is absent in these results.

```
          max_update_step="0.01" power="1.0" bias="0.0"
          refractory_period="0.0"
          time_membrane="1.0"/>
</zero_based_sigmoid_algorithm>
<positive_zero_based_sigmoid_algorithm
   name="exc_sigmoid_alg">
 <algorithm_parameter rate_max="100.0" noise="1.0"
          max_update_step="0.0005"
          power="1.0" bias="0.0"
          refractory_period="0.0"
          time_membrane="0.01"/>
</positive_zero_based_sigmoid_algorithm>
```

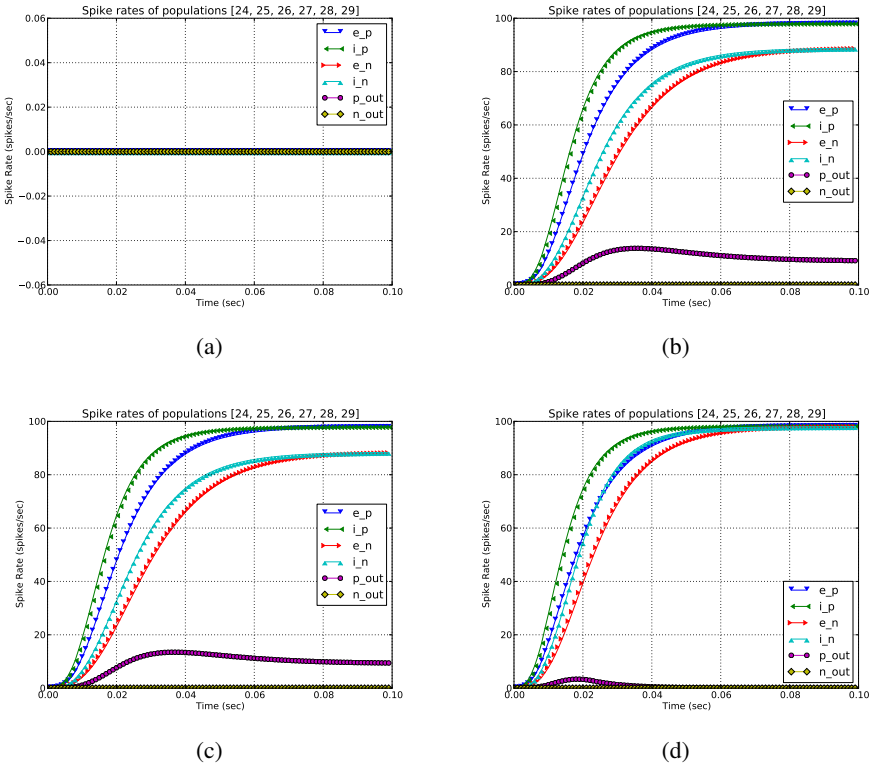


Figure 2.7: Plots of circuit activity for the output layer for the XOR-Network. (a) Input of 0Hz to each input layer population. (b) Input of 0Hz and 90Hz to the input layer populations. (c) Input of 90Hz and 0Hz to the input layer populations. (d) Input of 90Hz to both input layer populations. A description of each of the populations of this circuit is given in section 4.2.1.

```
  <positive_zero_based_sigmoid_algorithm
     name="inh_sigmoid_alg">
    <algorithm_parameter rate_max="100.0" noise="1.0"
           max_update_step="0.0005" power="1.0" bias="0.0"
           refractory_period="0.0"
           time_membrane="0.005"/>
  </positive_zero_based_sigmoid_algorithm>
</algorithms>
<circuit_descriptions>
  <circuit_description name="perceptron" number_of_roles="6">
    <circuit_node_role name="e_p"
           x_pos="3.0" y_pos="-2.0" z_pos="0.0"
```

```
            excitatory="True" output="False"
            input="True" positive="True"
            algorithm="exc_sigmoid_alg"/>
    <circuit_node_role name="i_p"
            x_pos="-3.0" y_pos="-2.0" z_pos="0.0"
            excitatory="False" output="False"
            input="True" positive="True"
            algorithm="inh_sigmoid_alg"/>
    <circuit_node_role name="e_n"
            x_pos="1.0" y_pos="-2.0" z_pos="0.0"
            excitatory="True" output="False"
            input="True" positive="False"
            algorithm="exc_sigmoid_alg"/>
    <circuit_node_role name="i_n"
            x_pos="-1.0" y_pos="-2.0" z_pos="0.0"
            excitatory="False" output="False"
            input="True" positive="False"
            algorithm="inh_sigmoid_alg"/>
    <circuit_node_role name="p_out"
            x_pos="3.0" y_pos="-2.0" z_pos="0.0"
            excitatory="True" output="True"
            input="False" positive="True"
            algorithm="exc_sigmoid_alg"/>
    <circuit_node_role name="n_out"
            x_pos="-3.0" y_pos="-2.0" z_pos="0.0"
            excitatory="True" output="True"
            input="False" positive="False"
            algorithm="exc_sigmoid_alg"/>
    <connection from="e_p" to="p_out" weight="2.0"/>
    <connection from="i_n" to="p_out" weight="-2.0"/>
    <connection from="e_n" to="n_out" weight="2.0"/>
    <connection from="i_p" to="n_out" weight="-2.0"/>
    </circuit_description>
  </circuit_descriptions>
 <dynamic_networks>
  <dynamic_network name="xor"
```

```
        dynamic_inputs="False"
        dales_law="True"
        use_bias="False">
  <layers>
    <layer>First</layer>
    <layer>Mid</layer>
    <layer>Last</layer>
  </layers>
  <circuit_names>
    <circuit>perceptron</circuit>
    <circuit>perceptron</circuit>
    <circuit>perceptron</circuit>
  </circuit_names>
  <ann name="xor"
      filename="WilsonCowanXORPerceptron.net"
      type="backprop">
    <algorithms>
      <algorithm>zero_based_sigmoid_n1_p1</algorithm>
      <algorithm>zero_based_sigmoid_n1_p1</algorithm>
      <algorithm>zero_based_sigmoid_n1_p1</algorithm>
    </algorithms>
    <backprop_trainer>
      <layers>
        <layer>First</layer>
        <layer>Mid</layer>
        <layer>Last</layer>
      </layers>
      <mappers/>
      <training_parameters learning_rate="1.0"
                           momentum="0.0"
                           use_batch="False"
                           use_bias="True"
                           relax="False"
                           max_epochs="10000"
                           training_set_mse="0.001"
                           validation_set_mse="0.0"
```

```
                generalisation_set_mse="0.0"
                log_file_name="log.txt"
                log_file_resolution="1"
                noise_level="0.0"
                noise_strength="0.0">
   <layers_to_train>
    <layer_name>*</layer_name>
   </layers_to_train>
   <filter_description type="trainingset_input_filter"
          input_layer_name="First"
          output_layer_name="Last"
          x_scale="1.0"
          y_scale="1.0"
          filename="None">
 <training_set>
  <training_units>
   <training_unit>
    <pattern height="2" width="1" num_features="1">
    0.0 0.0
    </pattern>
    <pattern height="1" width="1" num_features="1">
    0.0
    </pattern>
   </training_unit>
   <training_unit>
    <pattern height="2" width="1" num_features="1">
    0.0 0.9
    </pattern>
    <pattern height="1" width="1" num_features="1">
    0.9
    </pattern>
   </training_unit>
  <training_unit>
   <pattern height="2" width="1" num_features="1">
   0.9 0.0
   </pattern>
```

```
            <pattern height="1" width="1" num_features="1">
            0.9
            </pattern>
           </training_unit>
           <training_unit>
            <pattern height="2" width="1" num_features="1">
            0.9 0.9
            </pattern>
            <pattern height="1" width="1" num_features="1">
            0.0
            </pattern>
           </training_unit>
          </training_units>
          <generalisation_units/>
          <validation_units/>
         </training_set>
        </filter_description>
      </training_parameters>
     </backprop_trainer>
   </ann>
  </dynamic_network>
</dynamic_networks>
<dynamic_mappers/>
<simulations>
  <simulation name="sim" start_time="0.0" end_time="0.1"
           report_time="0.001" update_time="0.0001"
           step_time="1e-05" log_time="0.01"
           max_iterations="100000"
        progress_filename="WilsonCowanXORPerceptron.progress"
        output_filename="WilsonCowanXORPerceptron_sim.data">
   <description/>
   <input_patterns>
    <input_pattern network="xor" layer="First" time="0.0">
     <pattern height="2" width="1" num_features="1">
     0.9 0.9
     </pattern>
```

```
      </input_pattern>
    </input_patterns>
  </simulation>
 </simulations>
 <visualisations>
  <visualisation name="vis1" simulation="sim">
   <description></description>
   <parameters x="0.0" y="0.0"
           high_colour="16711680" mid_colour="16777215"
           low_colour="255" background_colour="15790320"
           high_value="1.0" mid_value="0.0" low_value="-1.0"
           zoom="5.0" frame="0" frame_step="1"/>
   <layers>
    <spatial_layer net="xor" layer="First"
              x_pos="0" y_pos="0" scale="10.0"
              layout="horizontal" show_ann="False"/>
    <spatial_layer net="xor" layer="Mid"
              x_pos="75" y_pos="0" scale="10.0"
              layout="horizontal" show_ann="False"/>
    <spatial_layer net="xor" layer="Last"
              x_pos="150" y_pos="5" scale="10.0"
              layout="horizontal" show_ann="False"/>
   </layers>
  </visualisation>
 </visualisations>
</project>
```

Chapter 3

Modelling Feature-based Attention with ANNs

This chapter discusses modelling Feature-based Attention using ANNs.

3.1 Learning for Feature-based Attention

From section 1.1, an image of the visual brain emerges as a distributed storage medium, with no single area maintaining a representation of objects as a whole, but as compositions of their constituent features. This distributed representation allows greater number of objects to be detected than a localized representation, which would have to store every combination of features for each object category, but at the cost of more complex 'recall' of objects by engaging each associated feature with the visual stimulus and using the presence and spatial relationships between features to determine the existence of an object category in the visual field. This linking together of disparate features to form a whole is termed 'binding'. In both CLAM and DIANNA the model comprises a distributed representation of form as V1 feature layers sensitive to lines of orientation. By using the V1 layer as a common 'blackboard' [18, 76] for the ventral forward and reverse neural streams, these distributed form features are bound together to allow the presence of the object as a whole to be determined. Maintaining object representations in a distributed fashion, rather than a Gestalt notion of a complete object, allows an

object to still be detected when presented with occlusion by other objects, albeit at a reduced strength of neural activity.

This idea of binding featural components at stimulus presentation in CLAM has been incorporated into DIANNA. However, DIANNA extends this idea to use the additional feature of colour.

3.1.1 Differences between feature types: form versus colour

In line with de Kamps and van der Velde's argument [18] for a combinatorial representation of real-world objects from separate features allowing novel objects to be represented through binding of existing neural features, it would seem appropriate that a mechanism for binding simple form features into complex objects would provide similar efficiency gains. For this reason a different neural architecture is proposed for dealing with form, where neural activity from simple V1 feature detectors undergoes more complex transformations in higher layers of the ventral stream, than colour, which would require no such transformations.

3.1.1.1 Colour feature detection

In DIANNA colour is modelled as a simple hierarchy of ventral stream layers (V1, V2, V4, PIT, AIT), with each colour stream for red, green and blue segregated from each other. The receptive fields of higher layers are mapped to predecessor layers via Gaussian-shaped receptive fields, with higher strength connections at the receptive field centre, and low strengths near the edge. No learning is performed on this architecture, reflecting the simple nature of colour processing. Although it should be noted that this is a simplification: in reality colour processing includes such operations as colour constancy and luminance adjustment [61, 37].

3.1.1.2 Form feature detection

DIANNA models form feature detection in almost the same way as CLAM [78]. The mapping of lower layers to higher layers is initially determined via the same Gaussian receptive field structure as in colour detection. However, inputs to V1 for each form feature have the real and imaginary components of Gabor filters

applied in opposition: the real component is applied to V1 as positive input, and the imaginary component as real valued negative input. In turn, orthogonal V1 form features input into V2 features, such that the V1 feature for 0° is input to a V2 feature as positive values, and the V1 feature for 90° is input to the same V2 feature as negative values. Likewise V1 features for 45° and 135° are input to a separate V2 feature. These two V2 feature layers feed into a common (for form) V3 layer.

This mixing of form features allows binding of simple features (orientation of lines) to be performed in higher layers, and for learning to occur. Clearly the process of learning will change the weights between layers, breaking the initial Gaussian receptive field structure of the form network. However, DIANNA departs again from CLAM in that no learning is performed in the forward network. Learning is performed instead purely in the reverse network, mapping AIT nodes to all lower layer nodes with weights such that the reverse nodes' activities match those of the forward layer when the stimulus and attentional template are both present. This method of learning is fast, as there are no hidden layers, allowing a simple delta-rule gradient descent to be used as the learning algorithm, and allows learning new attentional templates by adding new reverse AIT nodes without destroying the weights from previously learned training sets.

3.2 Neural Training

3.2.1 Network-based feature learning

In CLAM the ventral stream is modelled with layers V1, V2, V4, PIT and AIT. V1 consists of a number of feature detectors for lines of different orientation. AIT consists of a number of populations representing each object type to be recognized from its component straight lines. The CLAM approach to feature learning, the network learns to associate V1 input to AIT output, but this is not achieved in a well defined way. That is to say that the trained network does not yield layers which perform as detectors of defined features; the representational knowledge is truly distributed throughout the network.

The problem with this approach to learning features is that the literature shows strong evidence for a separation of responsibility between each layer, which can-

not be demonstrated within networks trained with this method. Furthermore, the inclusion of more feature detectors only increases the crosstalk between the existing feature detectors. This also assumes that all features are detectable equally strongly in all objects presented at all locations. This requires care to be taken in the development of training patterns.

Training a five layered feed-forward network is also problematic. Backpropagation of errors through four layers is computationally expensive. If objects are learned at all scales, different features may be detected as salient at each scale, requiring additional neural representations to be stored. Also if objects are not presented to the network with care, artificial features caused by the presentation may be learned. Boundary effects must also be considered if complete objects need to be presented during training.

3.2.2 Layer-based feature learning

An alternative to training the network as a whole is to train each layer separately on the inputs of its predecessor. This learning does not require backpropagation, as training an individual layer removes the 'hidden' nodes that motivated the need for backpropagation [58], and a simple delta learning rule will suffice.

In this approach, V1 inputs to V2, which is then trained to respond to points of intersection explicitly, or to learn other features of V1. V4 receives the output pattern of V2 as input. V4's output pattern is constellations of V2 features. In turn this output pattern is used as input to PIT, and PIT to AIT, using the same simplified supervised training mechanism of reducing errors via a delta rule (i.e. by determining the error between desired and actual activity, and adjusting the weights to reduce this difference).

The role of each layer can be more easily defined with this new approach:

- V2: Detects conjunctions of simple V1 features, but the features detected by V4 and above are less clear:
- V4: Scale invariance is achieved/mediated by not outputting joined up shapes from V2. If V4 also receives input from V1, then its role could be to learn cohesive shapes. V4 is also strongly linked to colour perception, itself another strong binding feature. V4 then learns which V2 shapes are bound together,

through learning binding relationships, potentially collaborating with colour perception binding information from V4. In addition, by coalescing V1 and V2 output streams, circles can still be detected as areas of lines without corners.

- PIT: Receives cohesive shapes from V4, and outputs learned associations. Its output pattern is a halo around all coherent components.

- AIT: Represented as clusters of nodes which together encompass the whole visual field. Each member of the cluster has a receptive field over part of PIT. This allows features and objects from cluttered visual input to still be resolved (overlaps excluded).

These features detection layers, with clearly defined responsibilities, will greatly improve the feature matching ability of the network, and make the network's behaviour considerably easier to understand. While these roles are arbitrary, they must be performed somewhere along the visual pathway, with the ventral stream being the most likely locations for each operation.

From a computational point of view, training a network this way is less intensive as there are no hidden layers, so backpropagation is unnecessary. However, the generation of training data becomes more of a challenge. Feature detectors for each of the various stages can be borrowed from the Computer Vision community to provide suitable target patterns for the supervised learning of each layer.

A similar approach has been taken by Hinton [31] to learn hand-written digits. Hinton's approach used a layered system of Restricted Boltzmann Machines using layers of either stochastic binary neurons or real-valued 'probability' neurons. The network is trained in a layer-by-layer approach to learn generative weights from the layer above. The intermediate layers of binary neurons are often referred to as 'feature detectors', while the top two layers of real-valued neurons form an associative memory. After training, the network may be fine tuned via backpropagation. Hinton termed this type of network a Deep-belief Network (DBN).

The construction of networks as layers of 'Nodes', defining connectivity between layers, and backpropagation were existing functions of DIANNA. Adding DBN functionality to DIANNA has simply required the addition of an additional 'Node' type and training algorithm.

3.2.3 Reciprocal network

The neural input progresses from V1 to AIT in a feed-forward fashion. Each layer therefore represents transformed neural "views" of the stimuli entering the eye. In order to attend to a particular feature, these neural views need to be matched against some form of attentional template. Two approaches to generating this attentional template were evaluated.

3.2.3.1 Hebbian Learning

A reverse network is created with the same neural structure as the forward network, but each connection in the forward network is reversed. The weights of these connections are then trained via Hebbian learning: each pattern is applied to the forward network, and if cell A causes cell B to fire, the corresponding connection in the reverse network from B to A is strengthened by the activity of cell A in the forward network. This method is the same as that presented in CLAM [78].

Some problems exist with this approach: firstly, the magnitude of connection weights in the reverse network is dependent upon the number of input patterns activating the corresponding forward node, therefore in the general case the connection strength between two nodes in the reverse network does not match that of the forward network. A second issue arising from this training method is the effect of boundary neurons receiving less activity than central neurons leading to lower weights in the reverse network, which causes the order of magnitude of weights in the reverse network differing greatly across each neural layer. The high variability of the reverse connection weights makes it impossible to parametrize an algorithm to work across disparate layers, and even the same layer.

3.2.3.2 Active State Learning

In this method, the learning of weights in the reverse network is a straightforward matching of the neural output of each feed-forward node at a time a stimulus matching the attentional template is present in its receptive field. In order to propagate the attentional template from higher brain areas to these feature layers, the AIT neurons are directly connected to neurons in the reverse network matching each neuron in each forward layer.

Training this reciprocal network is straightforward, with the desired output being to minimize the global error between activity in reverse neurons and activity in their matching forward neurons, by calculating the error in activity between the nodes in the forward network and those of the active state learning network for each location. As there are no hidden layers in architecture, backpropagation is not required, so a simple delta rule can be used to update the weights between the reverse network AIT neurons and nodes in all other layers of the reverse network, with the target output activity of the reverse network nodes being the activity of the nodes in the same location of the forward network. This thesis terms a network trained in such a way to reflect the state of a forward network an "active state network".

The structure of the layers in the reverse active state network are identical to the layers of the forward network (or network ensemble) it was created from, in terms of the layer sizes and micro-circuit structures (for the dynamical networks – see chapter 4). However, as noted above, the inter-layer connection structure is entirely different: each AIT node is completely segregated from its peers, so new attentional templates can be learned without loss of previously learned templates. In terms of biological realism, this architecture is unlikely. However, this mechanism could be relayed via the thalamus, or simply propagated in a direct one-to-one mechanism from higher to lower areas (only one neuron per layer would be necessary to carry the selected attentional template) with an intermediary layer of neurons weighted to match the forward activities in each layer.

3.3 Object recognition

In [78], CLAM demonstrates object recognition in the forward ventral stream by having each AIT population correspond to an object category. The network is trained using the network-based approach (see 3.2.1) with backpropagation. However, the presentation of objects occurs at only four distinct (non-overlapping) locations, with four different objects. This means the neural network has only to learn to discriminate four patterns, and these weights could be learned at a single location and copied to the other three. This thesis presents objects at all locations, such that all object locations are exposed to significant overlap. The resulting interference from this overlap greatly reduces the network's ability to learn the object

categories. Additionally, learning a new object category requires the previously learned categories to be relearned (although convergence will be quicker for these already learned patterns). As more objects are learned by the network, the relative strength of activity of the correct AIT population versus the other AIT populations will be diminished. The number of objects able to be learned by this network is therefore restricted.

The focus of CLAM, and this thesis, is feature-based attention, not object recognition. As such, the object recognition problems identified are not addressed. However, a Self-Organizing Map (SOM) [36] can be trained from the layers of the forward network by mapping SOM locations to the ventral stream layers via an intermediate layered network with a receptive field connection structure to the underlying ventral stream layers. These receptive field inputs are summed to produce the input value to the SOM for each neuron at each location. The intermediate network has N features per layer, where each of the N features detects a different spatial frequency and has a different receptive field size. Each feature in a given layer of the intermediate network connects only to equivalent layer in the forward network (i.e. the intermediate layer for V4 pools activity of neurons within the forward network V4 only). In this way a codebook vector can be constructed from activities within the hidden layers of the forward network. Figure 3.1 shows this architecture graphically.

This architecture allows a SOM to be trained from a network by evolving the network to a given input pattern and using the generated state as training data to the SOM. The trained SOM clusters similar neural representations to spatially congruent neurons. A bank of 'grandmother neurons' could be linked to each learned cluster to provide object recognition in a spatially agnostic way. The use of a SOM allows new object categories to be learned without needing to re-learn previous categories, and will scale better than backpropagation of the forward ventral stream, as each layer is used in object recognition, rather than only PIT, as is the case with CLAM.

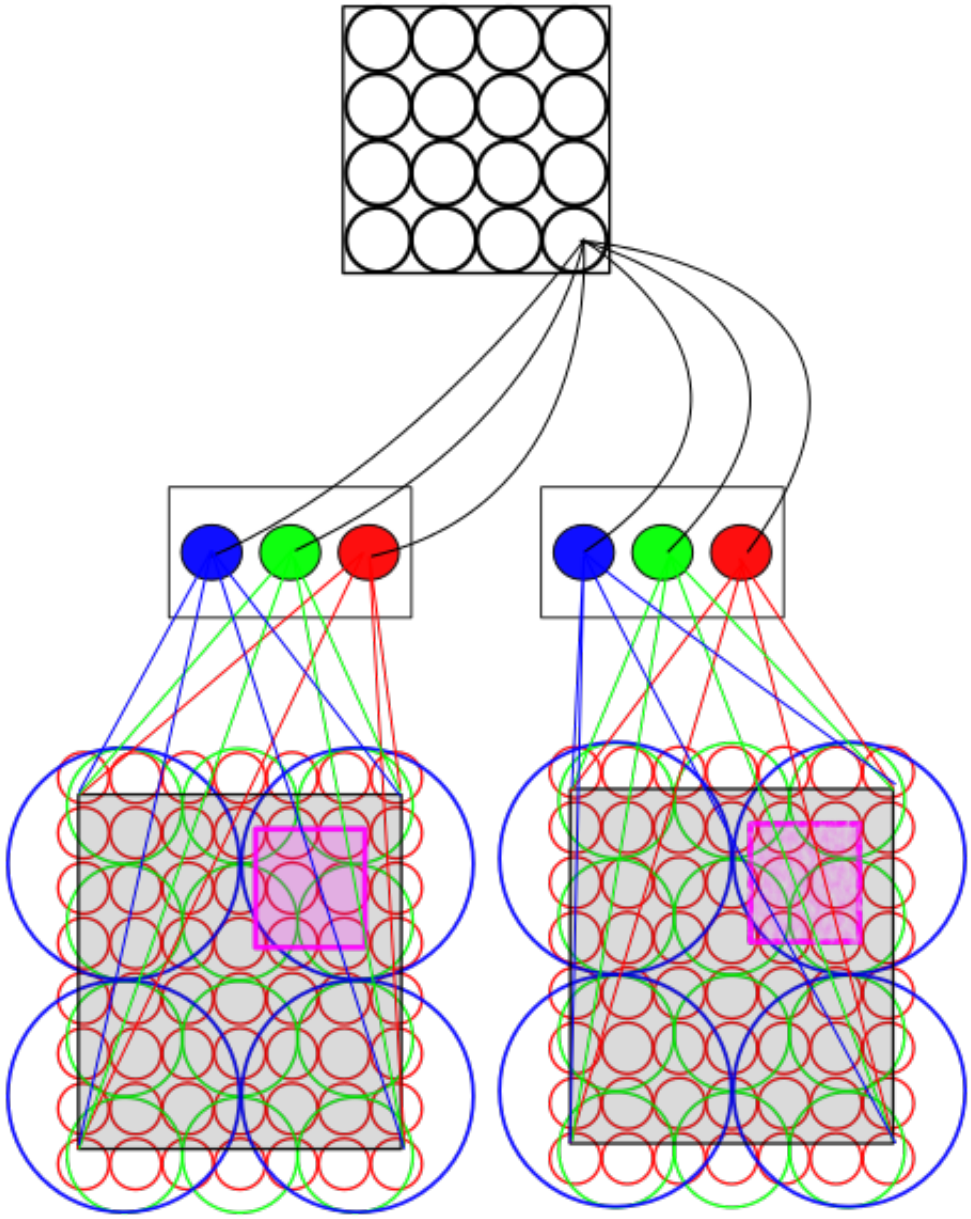


Figure 3.1: Connection structure of SOM (top) to two forward network layers (bottom) via two intermediate layers (middle). Every SOM node connects to every intermediary layer feature. Each intermediary layer pools neurons at distinct spatial frequencies only from the equivalent layer of the forward network. The codebook vector for the SOM is then the activities of all intermediary network nodes. Only connections from one SOM node are shown.

3.4 Visualization

The complexity of the visual system, particularly its implementations in CLAM and DIANNA, poses problems of scale both structurally and in terms of the volume of simulation data the model generates. Previously these simulations were checked by hand on test runs of small networks. This approach will not scale to the large number of neurons and fine time graining required to run accurate simulations. In order to more effectively check the accuracy of simulations, visualizations of the captured simulation data are necessary. The ability to watch activations flowing through the network in response to stimuli allows researchers to visually compare the simulation to their expectations and thus serves as a crude but expedient means to check their models' correctness. In addition, visualizations allow a greater understanding of the model when presented to researchers unfamiliar with the mechanism of visual search.

3.4.1 3D Network-based visualization

Prior to the development of DIANNA, early work was performed on extending CLAM and producing visualizations of the large-scale networks produced. An example 3D view of the CLAM model of the ventral stream is presented in figure 3.2.

The visualization leverages the hierarchical nature of CLAM to avoid overloading the viewer with unnecessary levels of detail, providing an overview first, zoom and filter, then details-on-demand as prescribed by Shneiderman [66]. As described previously, the model comprises two largely independent streams of neural networks, composed of two feed-forward networks for bottom up and top down activations.

Further complexity (and biological accuracy) is added by modelling the nodes of these feed-forward networks as populations of dynamical neurons (see chapter 4). The visualization reduces the complexity by allowing the user to select which details to view in terms of streams, layers within those streams, circuits within layers, and neurons within circuits. The part/whole relationship allows the use of transparency to semi-automatically manage the level of detail in a similar fashion described by Balzer and Duessen's Hierarchical Nets [3] for software visualization.

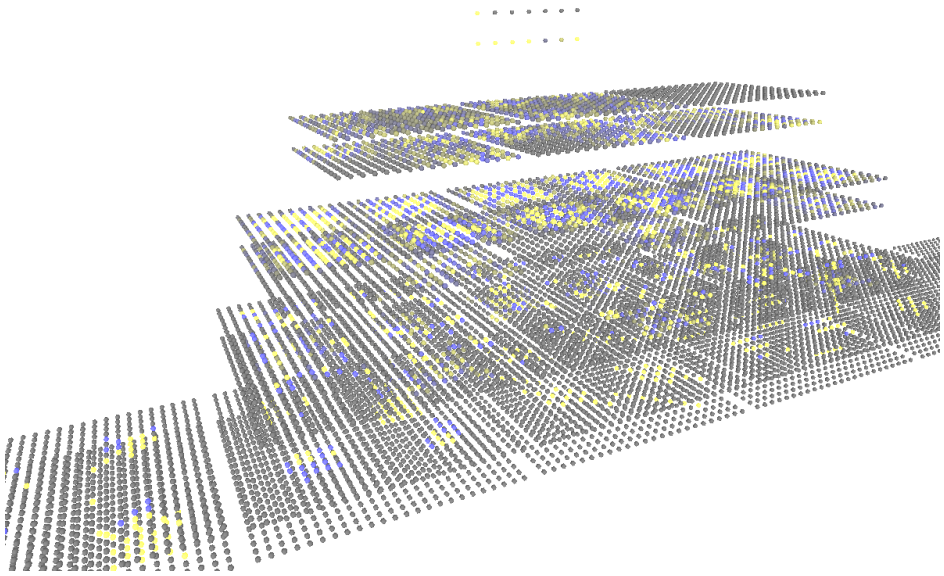


Figure 3.2: 3D View of CLAM ventral stream network. 5 pairs of layers representing V1, V2, V4, PIT, AIT (bottom to top), with the bottom layer of each pair representing the forward network, and the top layer representing the reverse network. Blue shows positive activity in neural populations, yellow negative activity, and grey no activity.

In particular, the use of transparency to manage the level of detail: that is, the artificial neurons appear as opaque spheres when the position of the viewer is more than a certain distance from them, otherwise the artificial neuron sphere is made progressively more transparent, allowing inspection of the dynamical nodes within (figure 3.3).

3.4.2 2D Layer-based visualization

DIANNA offers a number of visualizations of the component neural networks. The main visualization presents the network as individual neural layers. Multiple layers can be visualized simultaneously, and at arbitrary locations, to provide a global view of the complete network's behaviour. Visualizations are created as part of a DIANNA project, and are stored in the project's XML file. Each layer may be added independently of the others, and viewed as either a dynamical network layer, an ANN layer, or as a difference between activity in nodes of two other layers (dynamical or ANN layers). Furthermore, the XML describes the global visualization parameters, such as inputs, frames, camera position, necessary to load

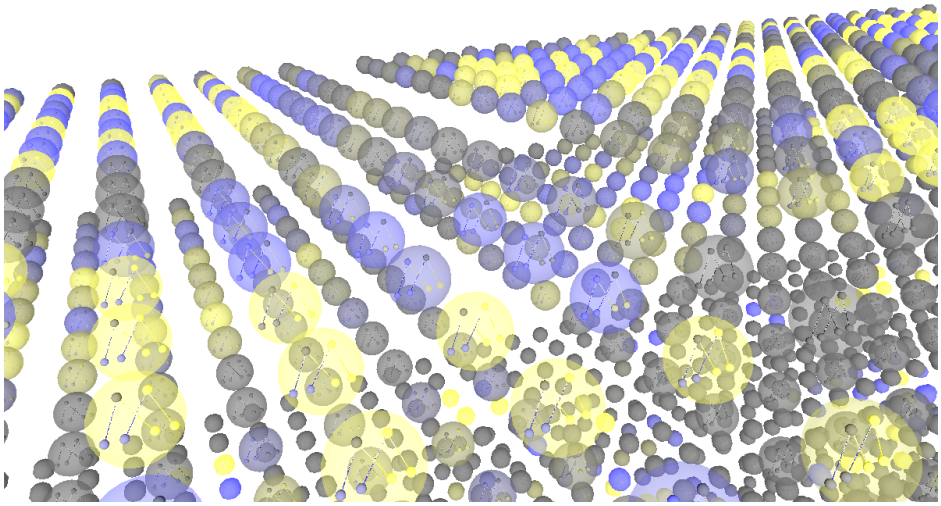


Figure 3.3: Close up of 3D View of CLAM. This shows the internal (dynamical) cortical circuits within a network for ANN nodes close to the viewer. Blue shows positive activity in neural populations, yellow negative activity, and grey no activity.

a visualization and generate an image from it unattended. This allows simulations to be rendered remotely, such as via a web-service.

Each layer has inputs and outputs, and DIANNA allows each of these to be visualized as a simple image. This will be particularly useful as the whole input-output chain of the neural network can be visualized directly, particularly the receptive field structure. The overlapping receptive fields naturally cause some blurring of the image at each subsequent stage, causing errors to grow through the network. A Gabor filter based approach allows each neuron's receptive field to be represented as weighted incoming activity, as opposed to a simple point activation as is performed in CLAM, and discussed below in sections 2.3.1 and 2.3.2. An image based approach to visualizing layers provides an easy means of inspecting how much this blurring obfuscates the input signal to AIT. Unless otherwise noted, all network images herein were produced via DIANNA.

3.5 Evaluation of Artificial Neural Networks for Feature-based Attention

A number of approaches to modelling feature-based attention with ANNs have been discussed in this chapter. In this section we introduce a common architecture for evaluating the different approaches described. Only the form feature modality of the ventral stream is evaluated, as the colour modality simply detects the presence of each colour, and its evaluation at the ANN level would not be informative. However, its performance is discussed in chapter 4.

It should be noted that the ANNs are not composed of cortical circuits. These are introduced into the model in chapter 4.

3.5.1 The Model

The model consists of two ANNs, one modelling the bottom-up flow of stimulus activities and the other modelling the top-down flow. The bottom-up network is a feed-forward network of five layers corresponding to V1, V2, V4, PIT and AIT visual areas. V1 consists of 4 feature layers which detect lines of 0°, 45°, 90° and 135° orientations, and objects are presented via input filters (see section 2.3) in the appropriate feature layer to simulate neural inputs from the LGN. The LGN is not included in the model.

A widening receptive field in higher layers allows AIT neurons to project across the entire V1 layer, allowing objects to be recognized in all locations. Each layer receives input from overlapping neighbourhoods of neurons in its preceding area and outputs to the next higher area such that the effective receptive field size increases towards higher visual areas. Evidence for the increasing receptive field structure from V1 to V2 is provided by Sincich and Horton [68]. The forward network is initially seeded with random weights connecting higher layers to their predecessor layers with circular receptive fields. The network is then conditioned using backpropagation with a low learning rate (0.1) for a small number of iterations (5): input/output pattern pairs are iteratively applied to the network, output errors are propagated back to the input layer, and network weights and neuron biases are adjusted to reduce the error until a threshold is reached. Each neuron in the network sums its input and applies this summed input through a sigmoidal squashing

function to determine the output of the neuron.

Once the forward network has been prepared it is used to train the top-down 'Active State' network (see section 3.2.3.2): each training pattern is evolved through the forward network and conditions reciprocal connection weights in the reverse network. This mechanism creates the attentional template. Note, that although the forward networks are weakly trained, training is not actually required to evaluate the performance of the reverse network, as the reverse networks is simply trained to match the activities of the forward network for each pattern.

The architecture of both forward and reverse networks is:

- V1: 4 input form features of 40×40 neurons.
- V2: 2 layers of 40×40 neurons.
- V4: 1 layer of 40×40 neurons.
- PIT: 1 layer of 40×40 neurons.
- AIT: 1 layer of 4×1 neurons corresponding to each shape.

Orthogonal angles from V1 project to the same layer of V2, but their weights are in opposition: V1 features for 0° and 90° project to one V2 feature as positive and negative weights respectively, and V1 features for 45° and 135° to the second. Each neuron receives activity from predecessor neurons in its receptive field, which are summed and the result is passed through an activation function:

$$o = f(\sum_i w_i x_i) \tag{3.1}$$

where o is the activity of a node, w_i is the weight of its i^{th} input and x_i is the activity of another node which is connected to the i^{th} input of this node. A `SigmoidAlgorithm` is used for the ANN, as this allows the sigmoid function, $f(x)$, to be smooth and is defined as:

$$f(x) = \frac{2}{1 + e^{(-\beta * (x - \text{bias})^{\text{power}})}} - 1 \tag{3.2}$$

β is a noise parameter, which alters the shape of the sigmoid function. The *bias* parameter allows the function to be moved along the x-axis. The *power* parameter

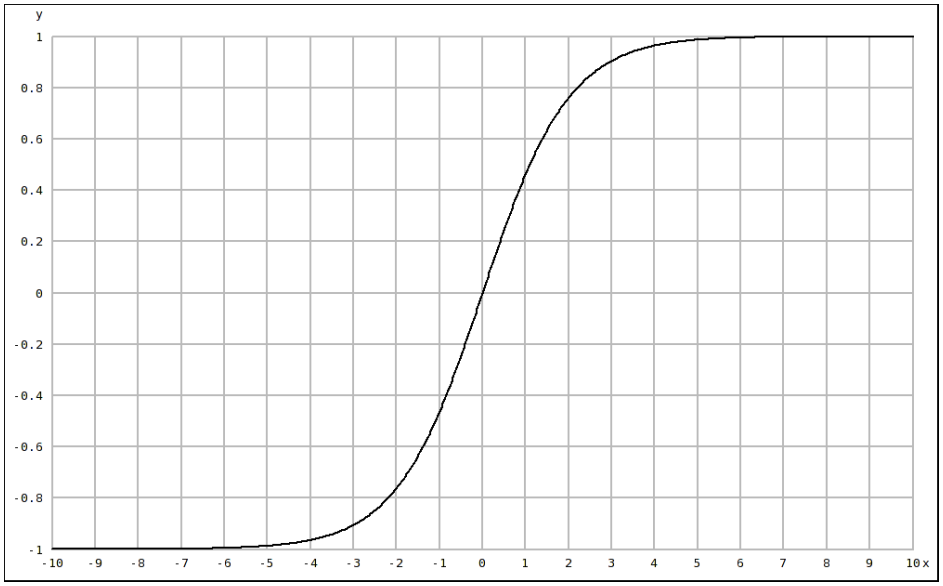


Figure 3.4: Plot of the `SigmoidAlgorithm` from eq. 3.2 with *bias*=0, β=1 and *power*=1

allows the function to have inflection points about 0. In this particular case, it means that for $x \to -\infty, f(x) = -1$ and $x \to \infty, f(x) = 1$. This form of the sigmoid is adjusted such that 0 input activity produces 0 output activity, as shown in figure 3.4. The *bias* term allows the sigmoid to be moved along the x-axis, and is not the same as the neuron's threshold, which is simply deducted from the input (x) prior to the activation function. While the effect is the same, the *bias* term is a property of the algorithm, and is not affected by training.

The application of a pattern to the input layer of the forward network causes activations to be propagated through the intermediate layers where form features are combined into successively more complex units, terminating in AIT. Consequently the forward pass loses spatial information with each ascent to a higher visual area, where the neurons are spatially unaware but activate strongly for the existence of the features to which they are tuned occurring anywhere within their receptive fields. Conversely, activation of the AIT neurons in the reverse network generates a cascade from the feature domain to the spatial domain as each area passes activations to lower visual areas, with greater retinotopic fidelity. A corollary to the AIT nodes being spatially agnostic is that the reverse network receives neural activity throughout the lower visual areas.

3.5.2 Evaluation Method

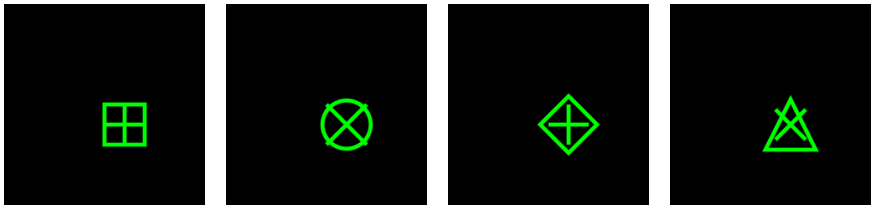
The neural inputs to the network are obtained from 2-dimensional log-Gabor filters with spatial frequencies of 0.071, 0.101, and 0.143 (3 decimal places), orientations 0°, 45°, 90°, and 135° with receptive fields of 10×10 pixels. The Gabor filters were applied to a 240×240 pixel image, as depicted in figure 3.5. The V1 network consisted of 4 features (1 for each orientation), each 40×40 neurons in size. Therefore each input neuron's receptive field overlapped its neighbours' receptive fields by 6 pixels. For each orientation, the real component of the output of the Gabor filter was applied as input to the network as a positive value, and the imaginary component was applied as a real-valued negative input value. The use of the imaginary component increases the amount of information extracted for each edge detected, as it shows the transition (dark to light, or vice versa), slightly offset from the real component. This mechanism then provides different inputs to the network when an edge within a neuron's receptive field is, for example, the left or right edge of a solid shape.

The reverse (attentional template) network's AIT node associated with each of the learned types is directly provided an input of 0.75 (the value input for each category during training of the reverse network). To generate statistics of the quality of matching between the forward and reverse networks for template stimulus pairs, a 'difference network' was created. This network calculates the difference in activity between each co-located neuron in the forward and reverse network for those forward neurons whose activity is not 0. The numbers presented are the absolute value of the total activity of this difference network for each combination of stimulus and attentional template, divided by the total activity in each layer of the forward network. This division was used as the activity of the forward network drives the total inhibition in the dynamical network presented in chapter 4. No activation function was applied to the output of neurons of the difference network.

Three experiments were performed with network inputs obtained from different components of the Gabor filters:

i). Untrained forward network with the real component of Gabor filters applied to V1 forward network.

ii). Untrained forward network with the real component of Gabor filters applied to V1 forward network and the real component of Gabor filters from lines



(a) Square. (b) Circle. (c) Diamond. (d) Triangle.

Figure 3.5: Images of the 4 shapes presented to the network. Each shape is presented individually to the network. The colour of the shape is not considered during this evaluation. The cross within the centre of each shape is to distinguish squares from diamonds, as otherwise, one is a 45° rotation of the other. It also serves to provide richer input to the network.

orthogonal to each V1 filter applied as negative input.

iii). Untrained forward network with the real component of Gabor filters applied to V1 forward network and the imaginary component applied as negative input.

In all cases the forward networks were untrained. The reverse network is an 'Active State' network trained for 20 epochs against the forward network, with a learning rate of 0.15. The reverse network was trained without momentum, and both connection weights and bias/threshold of the reverse ANN nodes were updated on-line (after presentation of each training pattern).

The sum of differences in activity between forward and reverse network locations is used a metric of how well the reverse network has learned the activities of the forward network for each object type. Low values show a good correlation between forward and reverse, while higher values show a poor match.

3.5.3 Results

The results are presented separately for each evaluation. The Appendix presents the raw data in tabular form. The following plots show the the sum across the entire network of mismatched activity between nodes in the same location of the forward and reverse network, for layers V2, V4 and PIT, divided by the total activity in the forward network, for layers V2, V4 and PIT. Only layers V2, V4 and PIT are used for these results, as the disinhibition network introduced in chapter 4 operates only

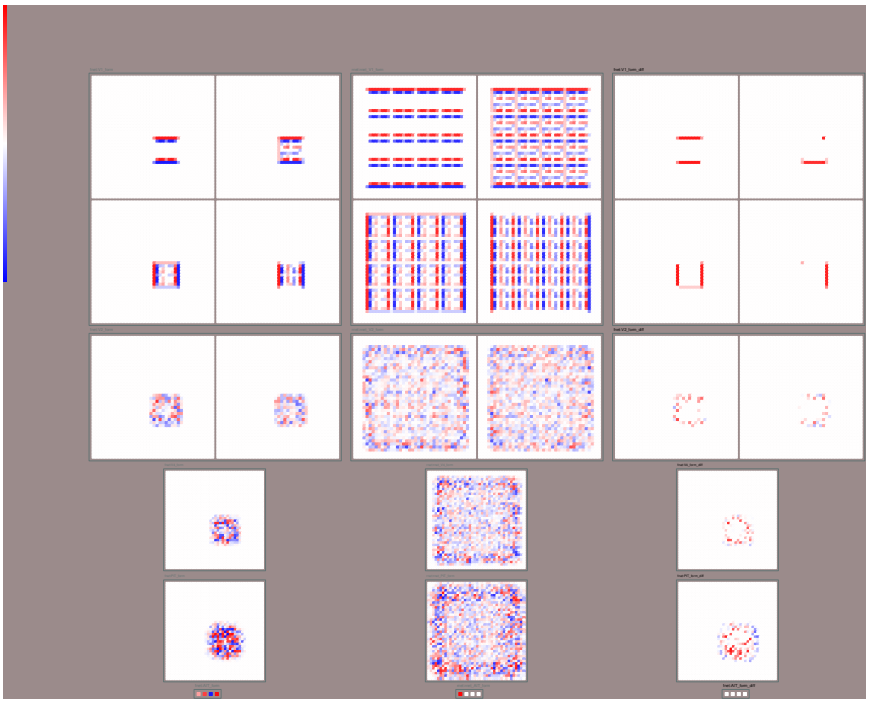


Figure 3.6: Example ANN for the evaluation. Each vertical network contains (from the top) V1, V2, V4, PIT, AIT. V1 contains 4 input features: a central square (see figure 3.5) is input to a Gabor filter bank, with the real component of the Gabor filter applied as positive input, and the imaginary component as negative input, for each of four orientations (clockwise from top-left: 0°, 45°, 90° and 135°). AIT consists of 4 features, for lines of each orientation. The left hand column shows the forward network, the central column the reverse network, and the right hand column the difference between them. Positive activity in the forward and reverse network is shown in red, with negative activity shown in blue. The right hand difference network shows red when the forward is positive and the reverse negative, and blue when the forward is negative and the reverse is positive. The difference network does not output when the sign of activity in the forward and reverse network agree, or if the node in the forward network has no activity. The response has been artificially enhanced for print.

on these layers. An example of the ANN is shown in figure 3.6.

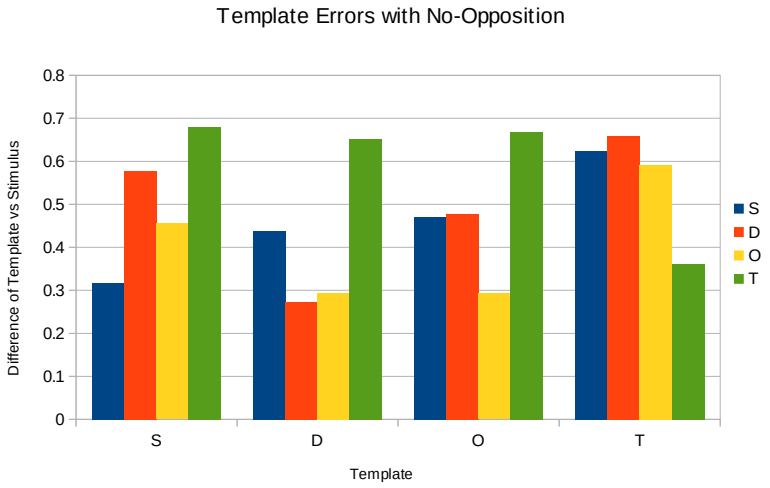


Figure 3.7: Results of i). Forward network with the real component of Gabor filters applied to V1 forward network. S=Square, D=Diamond, O=Circle, T=Triangle.

3.5.4 Discussion of results

When the reverse network is trained with images in all locations the training patterns are forced to overlap. This dramatically reduces the degree of matching between the input stimulus (in the forward network) and the attentional template (in the reverse network). In order to improve the matching between forward and reverse networks it is desirable to have greater heterogeneity in the input stimulus for each input pattern. The results shown in figure 3.7 (and tables A.1, A.2, A.3 and A.4 in the appendix) demonstrate poorer matching between stimulus and attentional template that the two other conditions (note the y-scale is smaller on figure 3.7 that both other conditions), as the input patterns are predominantly a single level of input for the interior of the input pattern, so only the edges of the pattern can be used to discriminate between different patterns. The overlap of the training patterns (shown in the V1 of the reverse network in figure 3.6) occurs across the edges of each location, so the main location of interference between different locations is also the most 'information rich' component of the pattern to learn.

In order to increase the heterogeneity of each input pattern for experiment (ii), orthogonal Gabor filters were applied as negative inputs (the results are shown in

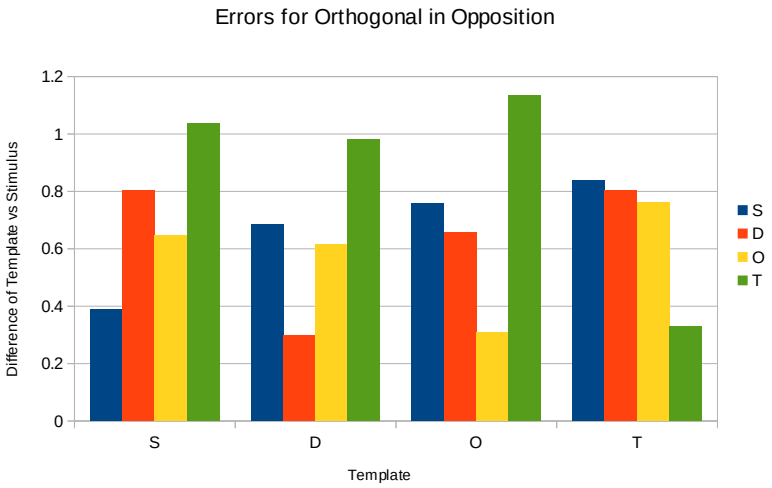


Figure 3.8: Results ii). Forward network with the real component of Gabor filters applied to V1 forward network and the real component of Gabor filters from lines orthogonal to each V1 filter applied as negative V1 input. S=Square, D=Diamond, O=Circle, T=Triangle.

figure 3.8 and tables A.5, A.6, A.7 and A.8 in the appendix). The output of these Gabor filters provide a marked increase in the variation of the input activity within each training pattern, leading to an improvement in the network's ability to discriminate between the input patterns for each template. This is likely caused by the positive and negative components summing to only a small activity at each input node. The small activity at the edges of the input patterns is further obfuscated by subsequent patterns presented in neighbouring locations which overlap.

The results for the third experiment (figure 3.9 and tables A.9, A.10, A.11 and A.12 in the appendix) show a considerable improvement over the previous results in terms of the magnitude of the difference of errors between template-stimulus matches template-stimulus mismatches. The results were obtained by applying the real and imaginary components of the Gabor filters in opposition (real component is positive input to V1; the imaginary component is negative input). Because the real and imaginary components of the Gabor filter are offset from each other, they are applied to neighbouring input nodes, and so do not suffer the same attenuation as experiment ii (applying orthogonal lines in opposition to V1). Moreover, this

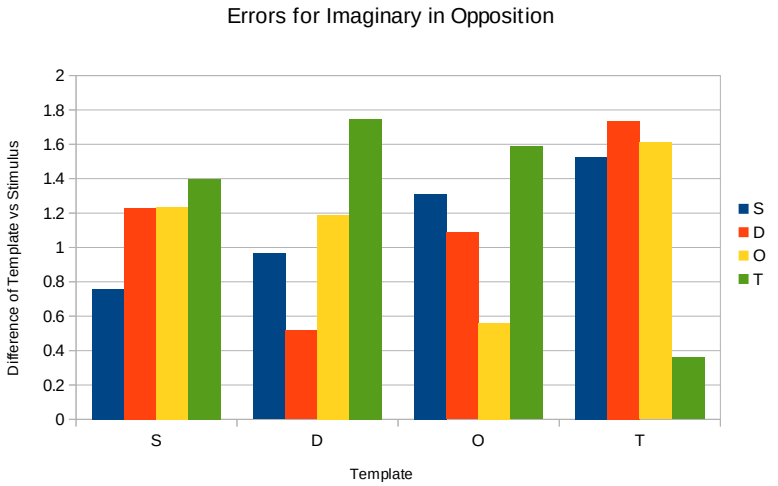


Figure 3.9: Results of iii). Forward network with the real component of Gabor filters applied to V1 forward network and the imaginary component applied as negative input. S=Square, D=Diamond, O=Circle, T=Triangle.

approach provides a different input for the left edge, versus the right edge or an input pattern, which improves the matching between neurons whose receptive fields are placed on the boundary of inputs. This approach leads to a greater variety in input activities (a more 'textured' input to V1). Therefore there is greater information provided to the network for each pattern, allowing the network to discriminate between input patterns more effectively, leading to a more accurate matching of the stimulus and attentional template.

Chapter 4

A Dynamical Model of Visual Attention

In this chapter, a dynamical model extending the ANN model introduced in chapter 3 is presented. The dynamical model is implemented in DIANNA, following the procedure in MIIND [16] described by de Kamps *et al* in [17], and discussed in chapter 2. The procedure detailed in [17] can be used to convert a biologically unrealistic ANN to more biologically plausible dynamical networks, with neural outputs being more accurately modelled with population firing rates [17, 40] using Wilson-Cowan population dynamics [82]. This conversion is necessary as the networks can only be trained as ANNs, due to the limitation of backpropagation and gradient descent requiring a partial ordering of the neurons (activities may only progress from layer N to layer N or $N+1$) [58]. The dynamical networks presented in this chapter are highly recurrent, so backpropagation is inappropriate for these networks. Also, the computational overhead of evolving these dynamical networks of partial differential equations makes their training computationally intractable, as network training requires the presentation and evolution of large numbers of training patterns. It should be noted that Wilson-Cowan dynamics uses "coarse time graining" as a means to convert these partial differential equations to ordinary differential equations [82].

Conversion of the ANNs to a dynamical system is necessary as ANN nodes do not correlate well with the properties of real cortical neurons [10]. Furthermore, ANNs only show the steady state of the network, as a consequence of their partial

ordering, whereas conversion of the ANN to a dynamical simulation allows the time-course of the system to be visualized and explored.

An additional consideration is that a population of spiking neurons can only have a positive spike rate, so the sigmoid function used in equation 3.2 is replaced by a positive only sigmoid function, where negative input activity is squashed to 0 output. In order to carry the negative 'signal', each ANN neuron is converted to a circuit of spiking neurons, in which some of the neurons convey the negativity of the input as positive rates, but their output is considered to be implicitly negative. The "perceptron circuit" described by de Kamps and van der Velde [17] is used as a basis for this conversion, and its function and embellishments from their work are described in section 4.2.1.

Figure 4.1: Plot of positive sigmoid algorithm from eq. 3.2 with *bias*=0, and β=1, where the output values are minimally bounded to 0.

4.1 Wilson-Cowan Dynamics

Wilson-Cowan dynamics [82] is a simplified model of the behaviour of a group of spiking neurons. The population firing rate of a group of neurons is the fraction of neurons that fire in a short time window, Δt, divided by Δt. Wilson-Cowan dynamics is given by:

$$\tau \frac{dE}{dt} = -E + f(\sum_i w_i E_i) \tag{4.1}$$

Here, E is the population firing rate of the group, E_i are the firing rates of other populations which are connected to the group via weighted connections i (with weight w_i). $f(x)$ denotes the sigmoidal squashing function applied to the sum of these weighted inputs. τ is the mean membrane time constant of this population. Although the original motivation for these dynamics has been criticized, the dynamics can also be inferred from sophisticated methods for modelling population dynamics and has recently been shown to reproduce neuronal dynamics very reliably in some cases [69].

Equation 4.1 gives a very direct interpretation for the activation in ANNs (as given by equation 3.1): if $\frac{dE}{dt} = 0$, equation 4.1 reads:

$$E = f(\sum_i w_i E_i) \tag{4.2}$$

i.e. if the sigmoids are identical in both equations, the equations are identical as well. The sigmoid in equation 4.1 arises from neuroscience considerations and will not be of the form of equation 3.2, but this is a minor issue.

This gives a direct interpretation for the activation of ANNs: they represent steady state activations of neural populations described by Wilson-Cowan dynamics. While equation 4.1 looks similar to equations describing Leaky-integrate-and-fire (LIF) neurons, LIF neurons describe discontinuous behaviour of individual neurons: the membrane potential is reset after a spike. It is therefore incorrect to associate these dynamics with population dynamics.

This suggests a direct possibility for converting ANNs into networks of dynamical simulations, following the procedure in [17]. Dynamical networks can be used to simulate networks of populations described by Wilson-Cowan dynamics. The most direct way of associating ANNs with neural dynamics is to generate Wilson-Cowan dynamical simulations from a trained ANN such that there is a one-to-one mapping between nodes of the ANN and the dynamical network. The weights in the dynamical network are the same as the weights between corresponding nodes in the ANN, with the exception of negative weights in the ANN mapping to negative roles in dynamic network circuits, and positive weights mapping to positive

roles. An example of the different positive/negative roles assumed by member populations of a circuit of the dynamical network is shown in figure 4.2.

4.2 Cortical Circuits

In the conversion from the ANN based model from chapter 3, each ANN node is replaced by a collection of dynamical nodes, representation populations of spiking neurons. This section discusses these collections as cortical circuits.

4.2.1 The Perceptron Circuit

The perceptron circuit is used in the conversion from an ANN to replace each ANN node, and is shown in figure 4.2. The roles of the circuit populations are:

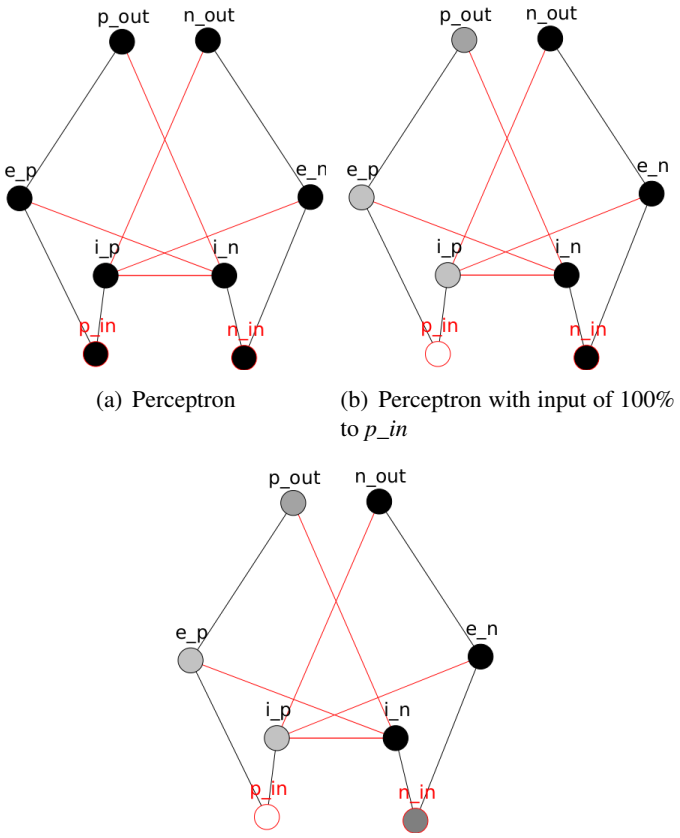
i_p (and i_n) Inhibitory input population (positive/negative): This population inhibits the negative input populations *i_n* and *e_n*, and the negative output population *n_out*. This population also receives inhibition from *i_n* when negative input is applied to the circuit, to provide a degree of filtering of noisy inputs.

e_p (and e_n) Excitatory input population (positive/negative): This population projects excitatory activity to *p_out* on positive input, and receives inhibition from *i_n* on negative input.

p_out (and n_out) Excitatory output population (positive/negative): This population is receives excitatory input from *e_p* on positive input, and is inhibited by *i_n* on negative input. This node (and its negative counterpart, *n_out*) may be considered as the output of the circuit.

The above descriptions of the roles of the named circuit populations is mirrored for the negative populations in the right-hand half of figures 4.2 (those populations whose names replace ‘p’ with ‘n’).

When converting the ANN to a population-based model a complication arises in the representation of negative ANN weights as the firing rate of a collection of neurons. This impedance was overcome by replacing each ANN node in the forward and reverse networks with a micro-circuit of six neural populations with four input



(a) Perceptron

(b) Perceptron with input of 100% to *p_in*

(c) Perceptron with input of 100% to *p_in* and 50% to *n_in*

Figure 4.2: The perceptron circuit: e_p - Excitatory positive input; i_p - Inhibitory positive input; e_n - Excitatory negative input; i_n - Inhibitory negative input; p_out - Positive output; n_out - Negative output; p_in - positive input; n_in - negative input. Red connections are inhibitory and black connections are excitatory. The nodes whose names are in red are external inputs to the circuit, and not part of the circuit itself. Node colour: black 0% activity, white 100%. A plot of the dynamics of this circuit is shown in figure 4.3.

nodes and two output nodes: one coding for positive ANN weights and the other coding for negative weights (see figure 4.2). Both of these nodes output positive spike rates, but the negative nodes' outputs are implicitly considered as negative values. The *p_in* and *n_in* nodes are not part of the circuit, but driver nodes to test the functionality. Positive input to *e_p* and *i_p* nodes ensures that the positive output population (*p_out*) of a circuit fires exclusively (see figure 4.3), as *i_p* inhibits activity of the negative output node, *n_out*.

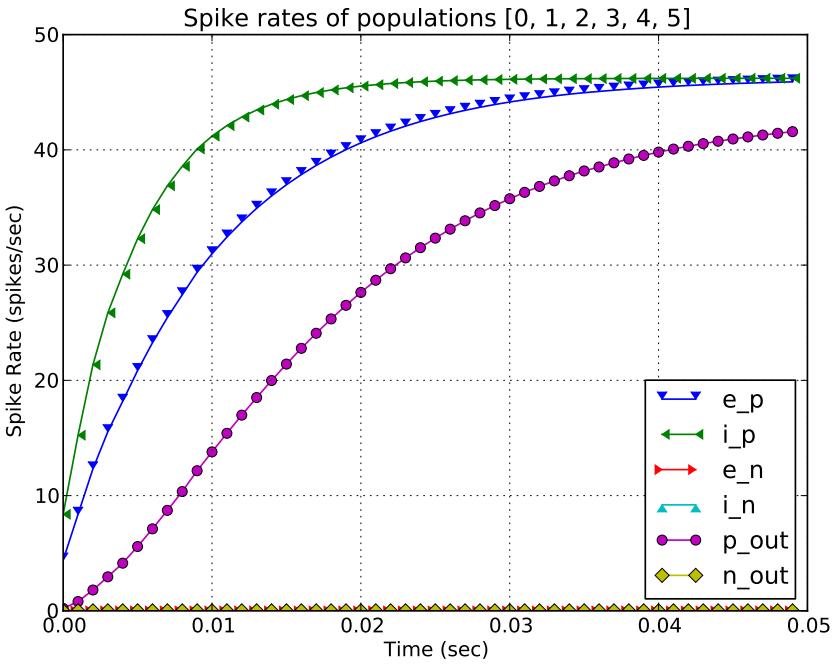
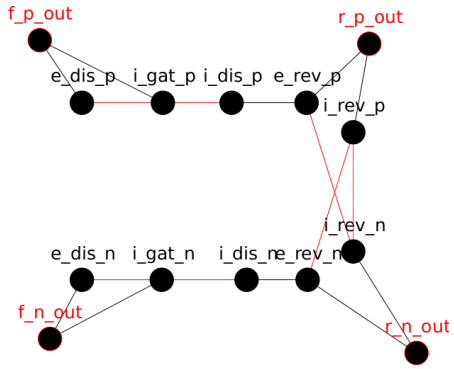


Figure 4.3: Plot of the dynamical behaviour of the perceptron circuit shown in figure 4.2. The connection weights from inhibitory populations was -2.0, and from positive populations 2.0. The circuit received a constant input spike rate of 100Hz (i.e. p_in from figure 4.2 was outputting spikes at 100Hz). The squashing algorithm used is a `PositiveZeroBasedSigmoidAlgorithm`, with parameters rate_max=100.0, power=1, noise=1. The excitatory populations (e_p, e_n, p_out, n_out) have a membrane time constant of 1×10^{-2}, and inhibitory populations (i_p, i_n) have a membrane time constant of 5×10^{-3}. Populations *e_n*, *i_n* and *n_out* are all 0 in this plot.

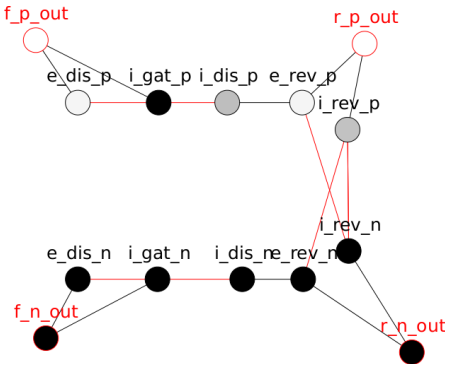
This circuit departs from that of [17] in that the *i_p* and *i_n* populations inhibit both negative or positive inputs respectively, as DIANNA connects layers by mapping receptive fields algorithmically using distance between nodes, whereas MIIND (as used by [17]) uses a more prescribed approach where connections are mapped between layers using stride lengths and receptive field sizes. The DIANNA approach is less precise than that taken by MIIND, but allows connections to be mapped between arbitrary sized layers. A consequence of this mapping is that perceptron circuits in DIANNA often receive both positive and negative input simultaneously. Another advantage of this cross inhibition is that recurrent connections in DIANNA may selectively inhibit positive or negative inputs, which may allow the output of

the perceptron circuit to switch the 'sign' of its output if the circuit is receiving both positive and negative inputs.

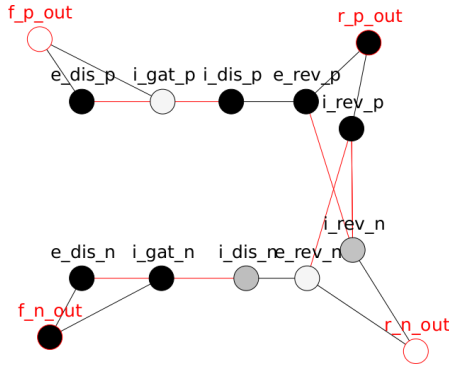
4.2.2 The Disinhibition Circuit



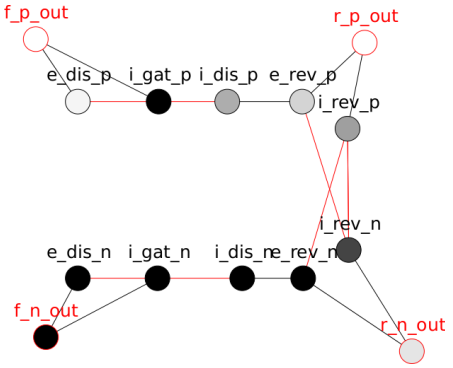
(a) Disinhibition with no input



(b) Disinhibition with input of 100% positive input from forward (f_p_out) and reverse (r_p_out) networks



(c) Disinhibition with input of 100% positive input from forward network (f_p_out) and 100% negative from reverse network (r_n_out)



(d) Disinhibition with input of 100% positive input from forward network (f_p_out) and 100% positive input from reverse network (r_p_out) networks and 90% negative from reverse network (r_n_out)

Figure 4.4: The disinhibition circuit. See text for explanation of circuit roles. Red connections are inhibitory, black connections are excitatory. The nodes whose names are in red are external inputs to the circuit, and not part of the circuit itself. Node colour: black 0% activity, white 100%. Plots of the dynamics of this circuit are shown in figures 4.5, 4.6 and 4.7.

The disinhibition circuit determines if there is matching of the activity in the forward and reverse networks in terms of whether both have the same sign (positive

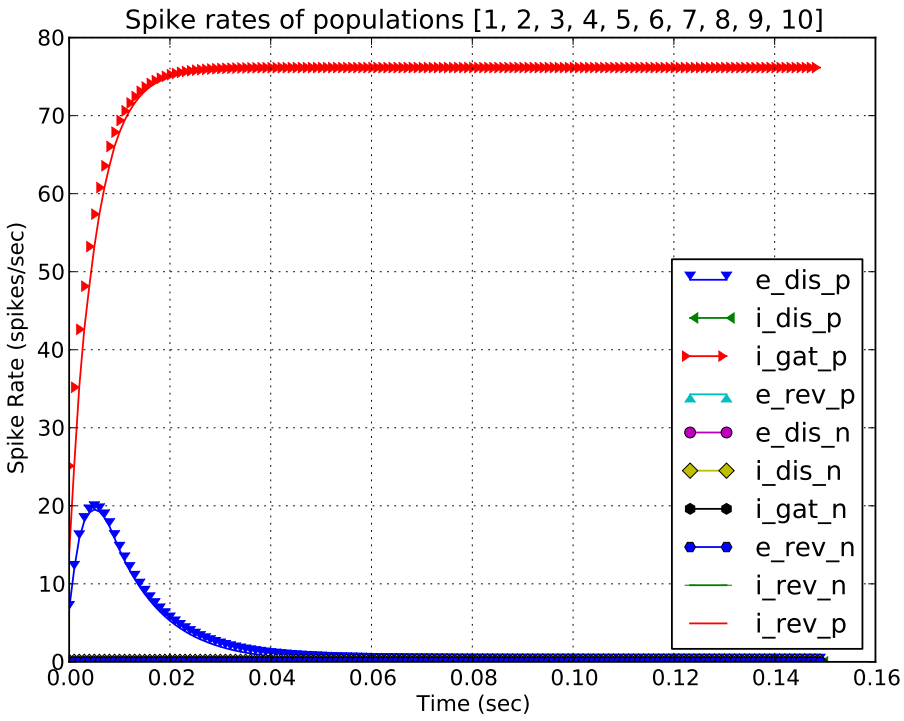


Figure 4.5: Plot of the dynamical behaviour of the disinhibition circuit shown in figure 4.4 with no input from the reverse network. The circuit received a constant positive input spike rate of 100Hz from the forward network (i.e. *f_p_out* from figure 4.4 was outputting spikes at 100Hz). The squashing algorithm used is a `PositiveZeroBasedSigmoidAlgorithm`, with parameters rate_max=100.0, power=1, noise=1. The excitatory populations (*e_***) have a membrane time constant of 1×10^{-2}, and inhibitory populations (*i_***) have a membrane time constant of 5×10^{-3}.

or negative). The strength of this agreement determines the degree of output of the disinhibition circuit, as either excitatory output in matching cases, or inhibitory output on mismatches. The member populations and their roles are:

e_dis_p Excitatory output population (positive): This population receives excitatory input from the forward positive population (*f_p_out* in figure 4.4). This population is active when the forward and reverse node activities match. This node (and its negative counterpart, *e_dis_n*) may be considered as the output of the circuit. When the forward and reverse networks do not match, this node is inhibited (to extinction) by *i_gat_p*.

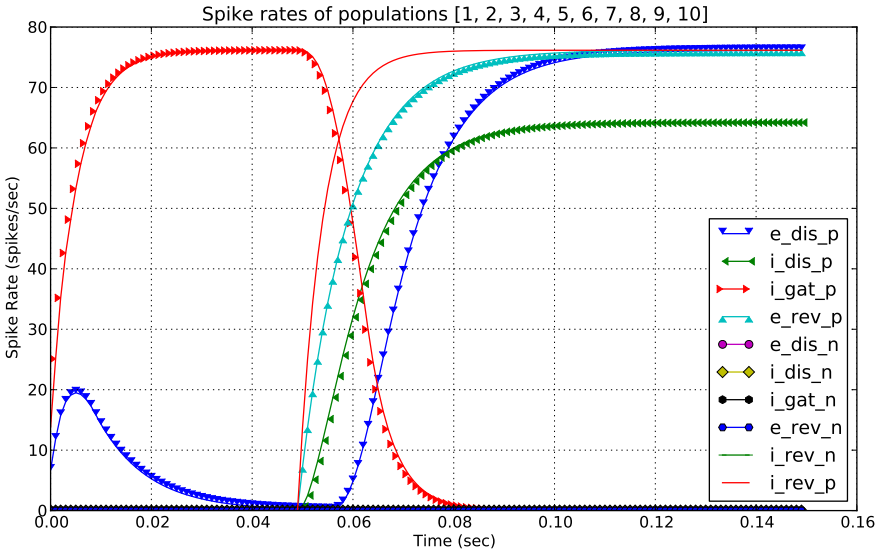


Figure 4.6: Plot of the dynamical behaviour of the disinhibition circuit shown in figure 4.4 with matching inputs from the forward and reverse networks. The circuit received a constant positive input spike rate of 100Hz from the forward network (i.e. *f_p_out*), and a positive input spike rate of 100Hz from the reverse network (i.e. *r_p_out*) from 50ms onwards. The squashing algorithm used is a `PositiveZeroBasedSigmoidAlgorithm`, with parameters rate_max=100.0, power=1, noise=1. The excitatory populations (e_*) have a membrane time constant of 1×10^{-2}, and inhibitory populations (i_*) have a membrane time constant of 5×10^{-3}.

i_gat_p Inhibitory gating population (positive): This population receives excitatory input from the forward positive population (*f_p_out* in figure 4.4), and inhibits the positive output. When the reverse network matches the forward, this node receives strong inhibition from *i_dis_p*, allowing output from *e_dis_p* (signalling a match between bottom-up and top-down networks).

i_dis_p Inhibitory disinhibition population (positive): This population is active when the reverse network has positive activity. It strongly inhibits the *i_gat_p* population, allowing *e_dis_p* to output if there is any input to *e_dis_p* from the forward network's positive population (*f_p_out*). In other words, this population *disinhibits* the *e_dis_p* population.

e_rev_p Excitatory reverse population (positive): This population receives excitatory activity from the reverse network's positive input (*r_p_out* in figure 4.4)

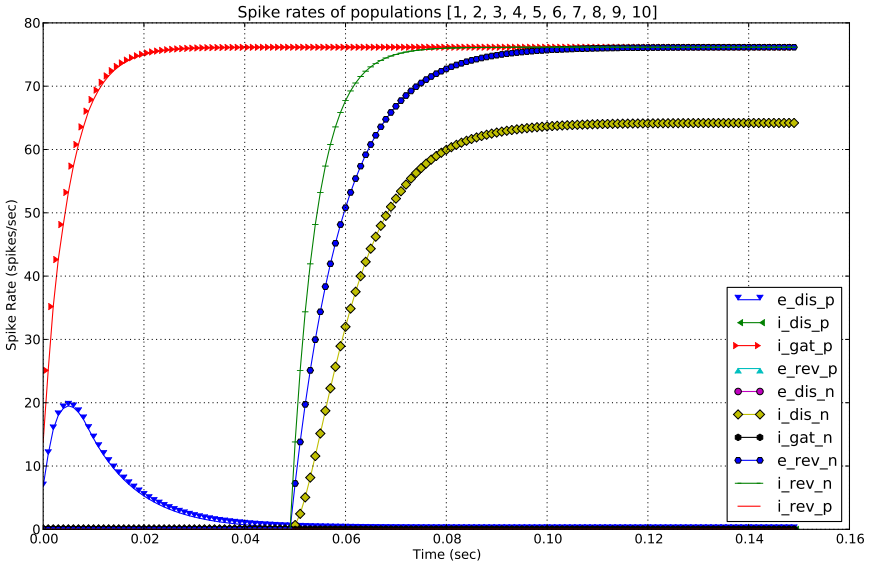


Figure 4.7: Plot of the dynamical behaviour of the disinhibition circuit shown in figure 4.4 with mismatched inputs from the forward and reverse networks. The circuit received a constant positive input spike rate of 100Hz from the forward network (i.e. *f_p_out*), and a negative input spike rate of 100Hz from the reverse network (i.e. *r_n_out*) from 50ms onwards. The squashing algorithm used is a `PositiveZeroBasedSigmoidAlgorithm`, with parameters rate_max=100.0, power=1, noise=1. The excitatory populations (e_*) have a membrane time constant of 1×10^{-2}, and inhibitory populations (i_*) have a membrane time constant of 5×10^{-3}.

and enables the disinhibition of *e_dis_p* by activating the *i_dis_p* population to inhibit the gating node (*i_gat_p*). Similar to the perceptron circuit, it also receives inhibitory input from *i_rev_n* to filter noisy inputs from the reverse network.

i_rev_p Inhibitory reverse population (positive): This acts as a filter for noisy input from the reverse network by inhibiting both *i_rev_n* and *e_rev_n*, so the circuit will only output the strongest reverse input (see 4.4 d)).

The above descriptions of the roles of the named circuit populations is mirrored for the negative populations in the lower half of figures 4.4 (those populations whose names end with 'n').

During the conversion to a dynamical network the forward and reverse networks

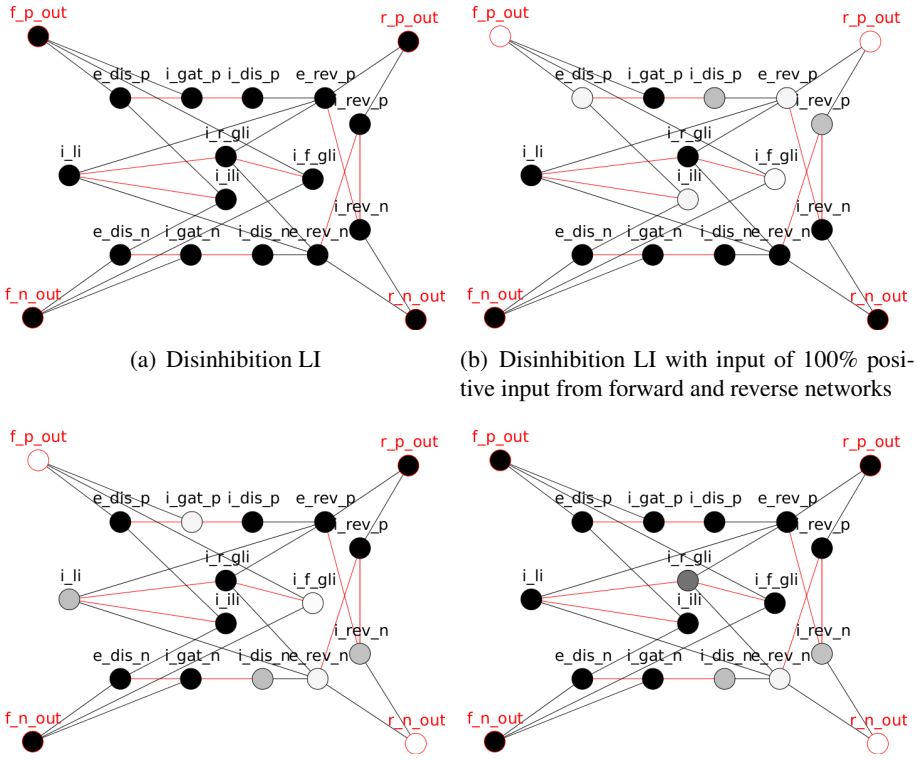
are coupled through another cortical circuit joining the positive and negative populations of the forward and reverse networks (see figure 4.4). The circuit acts to gate activations between the two networks through a disinhibition mechanism driven by matching activations in the joined forward and reverse networks. For a positive activation in the forward network (from *f_p_out)* the output and gating nodes of the disinhibition circuit (*e_dis_p* and *i_gat_p* respectively) receive activations from the forward network, such that the output node (*e_dis_p*) is excited and inhibited equally after a brief settling period in which the output of the gating node (*i_gat_p*) catches up with the innervation of the disinhibition circuits output node (*e_dis_p*, or *e_dis_n*), driving the spike rate of this population to 0 (see figure 4.5). This steady state is maintained by activations in the forward network sustaining the input to the circuit from *f_p_out*. If a positive activation occurs in the feedback network (node *r_p_out*), the disinhibition node (*i_dis_p*) is activated in the disinhibition circuit causing the gating node (*i_gat_p*) to be inhibited, allowing the output node (*e_dis_p*) of the disinhibition circuit to output spikes. This scenario is shown in figure 4.6, where matching (positive) input is applied to the reverse network from 50ms onwards. A similar mechanism occurs for matching negative activations in the forward and reverse networks. Note that in cases of mismatched activations, the output node of the disinhibition circuit will not emit spikes as the gating nodes (*i_gat_p* or *i_gat_n*) will not be inhibited by their disinhibition nodes (*i_dis_p* or *i_dis_n* respectively), as shown in figure 4.7.

4.2.3 The Disinhibition Circuit with Lateral Inhibition

The disinhibition circuit with lateral inhibition extends the disinhibition circuit with the addition of four new populations. The description of the populations common to this circuit and the disinhibition circuit without lateral inhibition are omitted for brevity, but are explained in section 4.2.2 above.

i_f_gli Forward gating of lateral inhibition population: This population receives excitatory input from the forward positive and negative populations, and inhibits the *i_r_gli* gating population, allowing lateral inhibition to occur when there is unmatched forward and reverse activity.

i_r_gli Reverse gating lateral inhibition population: This population receives excitatory activity from the reverse network to inhibit lateral inhibition. It is inhibited by the *i_f_gli* population when there is activity in the forward net-



(a) Disinhibition LI

(b) Disinhibition LI with input of 100% positive input from forward and reverse networks

(c) Disinhibition LI with input of 100% positive input from forward network and 100% negative input from reverse network

(d) Disinhibition LI with input of 100% negative input from reverse network

Figure 4.8: The disinhibition circuit with lateral inhibition. See text for explanation of circuit roles. Red names are input nodes, red connections are inhibitory. Node colour: black 0% activity, white 100%

work. The purpose of this population is to prevent lateral inhibition in areas with no forward activity, so that neighbouring populations which may have matching activity are not attenuated.

i_ili Inhibition of lateral inhibition population: When the disinhibition circuit receives matching input from the forward and reverse networks, this population receives strong excitatory activity from *e_dis_p* and *e_dis_n* populations, and inhibits *i_li* to extinction, preventing lateral inhibition when the stimulus and attentional template match.

i_li Lateral inhibition population: This population receives excitatory activity from the reverse network via *e_rev_p* and *e_rev_n*, so is only active when attention is deployed. However, it receives strong inhibition from both *i_ili* and *i_ili*

so that its activity is extinguished when there is no forward activity, or when the forward and reverse activity matches. When active, the *i_li* population inhibits its neighbouring cortical circuits' *e_dis_p* and *e_dis_n* populations.

Chelazzi and collaborators have demonstrated that when several features in AIT compete with each other, and one of them is task-relevant, then it will suppress the other features' activation. Subsequently feature-based activation is relayed to *all positions* in lower areas of visual cortex where it can interact with the visual stimulus information that is present there [10]. It has been shown [75] that consistency checks can be performed by local cortical circuits. These local circuits have a higher activation if the locally present visual stimulus information matches the feature that corresponds to the feature-based attention activation, i.e. when local bottom-up stimulus-driven information matches attention-driven top-down information. It has been demonstrated by van der Velde and de Kamps [75] and in experiment (iii) in chapter 3, that there are many more such local matches at the retinotopic position where the feature of interest resides, i.e. by matching top-down and bottom-up information it is possible to isolate the retinotopic position of the relevant feature in low visual areas, where position information is present, unlike in higher areas such as AIT.

Spurious matches at locations which do not match the attentional template do occur, as the response of each neuron can be only positive, negative or silent. In the forward network silent neurons generally occur at locations where there is no stimulus, which leaves neurons for locations with a distractor stimulus a 50% chance of matching the attentional template in the top-down network, as the network matches the *sign* of activity in the forward and reverse network, which can only be positive or negative if there is a stimulus within their receptive field.

Each neuron in the forward network responds to local activity received from predecessor layers in a bottom-up fashion, so there is no mechanism which allows these spurious matches to be detected as such by the responding neuron. Likewise, there is no mechanism which allows correctly matching neurons to determine they are part of a neighbourhood matching the attentional template. However, the degree of mismatches, where there is activity in both forward and reverse networks, but of different sign, is lower where the top-down network activity matches the stimulus-driven forward network. By using these mismatched populations to inhibit activity in neighbouring populations of the same layer (lateral inhibition) these areas of

spurious matches can be attenuated, which will reduce the propagation of activity from non-matching locations to higher areas of the ventral stream, or reduce the activity of those areas to the dorsal stream.

4.2.4 The LIP Circuit

Once the retinotopic position is found, it can be transferred to the parietal cortex (area LIP) to prepare a saccade, or local information can be reprocessed, for example, to discover which other features belong to this particular object as demonstrated in [18]. This is an interpretation of binding as a two-step process, rather than as a state [76].

The LIP circuit is a simple circuit of a single excitatory population receiving projections from *e_dis_p* and *e_dis_n* disinhibition populations from all disinhibition layers (V2, V4 and PIT).

4.3 A Dynamical Model

An ANN model of the ventral stream was created with the following layers:

The architecture of the network is:

- V1: 4 input form features of 40×40 neurons.
- V2: 2 layers of 40×40 neurons.
- V4: 1 layer of 40×40 neurons.
- PIT: 1 layer of 40×40 neurons.
- AIT: 1 layer of 4×1 neurons corresponding to each shape.

This network was trained using backpropagation for 5 epochs with 256 training samples of the 4 shapes from figure 3.5 in 16 overlapping locations. The inputs were applied from Gabor filters with line orientations 0°, 45°, 90° and 135°, with the real component of the Gabor filter applied as positive input to V1, and the imaginary component applied as negative input.

After training was complete, a complementary reverse network was trained from the forward network using the 'Active State' training algorithm discussed in section 3.2.3.2 for 20 epochs, with a learning rate of 0.15.

The trained ANNs was then converted to a dynamical network by replacing each ANN node with a perceptron micro-circuit, in the manner described by de Kamps and van der Velde in [17]. A disinhibition network with 3 layers of matching dimensions to V2, V4 and PIT of the previously trained ANNs with each 'node' being a disinhibition circuit of Wilson-Cowan populations was then created. The forward and reverse layers were connected to the disinhibition layer as described in section 4.2.2. In addition, a LIP layer was created with the same dimensions as the others. Both the ANN and dynamical networks used sigmoid algorithms (`ZeroBasedSigmoidAlgorithm` for the ANNs and `PositiveZeroBasedSigmoidAlgorithm` for the dynamical networks). For the the dynamical networks, the membrane time constant for excitatory populations was 1×10^{-3} and 5×10^{-4} for the inhibitory populations. Further parametrization of these networks is provided in the appendix (section A.4).

Two non-overlapping input patterns were applied to the forward network and allowed to propagate to AIT and the network was allowed to settle for a period of 150ms. At this time an attentional template matching one of the patterns was activated in the AIT layer of the reverse network.

In all images in the results sections, input to the form network was from Gabor filters with line orientations 0°, 45°, 90° and 135° applied to V1 features top-left, top-right, bottom-left and bottom-right respectively. The imaginary component of each Gabor filter was applied as negative input. Additionally for colour networks, input was applied via colour input filters for each of the colour channels red, green and blue.

For the form-only simulations, dynamical simulations were run for all combinations of shape, with shapes applied in non-overlapping locations. Only images of simulations with inputs of a square in the top-left, a triangle (bottom-left), diamond (top-right) and a circle (bottom-right) of the visual array are provided in the results section. For form-and-colour simulations, results are presented for an input array of the same four form objects, with a distractor object sharing the target object's colour.

4.3.1 Results: Form Network

Figures 4.9 and 4.10 show the activity of the dynamical networks 150ms after cue onset and 350ms after application of the attentional template for square, respectively. The network does not use lateral inhibition to reduce the effect of spurious (sparse) matches.

The visual array for this experiment is a square (top-left), triangle (bottom-left), diamond (top-right) and a circle (bottom-right).

4.3.2 Results: Form Network with Lateral Inhibition

Figures 4.11 and 4.12 show the activity of the dynamical networks 150ms after cue onset and 350ms after application of the attentional template. The network uses lateral inhibition to penalize mismatched activity in the forward and reverse networks.

The visual array for this experiment is a square (top-left), triangle (bottom-left), diamond (top-right) and a circle (bottom-right).

4.3.3 Results: Form and Colour Network with Lateral Inhibition

Figures 4.13 and 4.14 show the activity of the dynamical networks 150ms after cue onset and 350ms after application of the attentional template. The simulation includes colour channels to demonstrate the effect of searching for multiple features. To allow colour mixing, the colour attentional template is applied as negative green and blue and positive red, to match pure red colour in the stimulus. The network uses lateral inhibition to penalize mismatched activity in the forward and reverse networks.

The visual array for this experiment is a red square (top-left), green triangle (bottom-left), red diamond (top-right) and a blue circle (bottom-right).

Figure 4.9: Activity of the dynamical network with disinhibition after stimulus onset. The network on the left (vertically) shows the forward network carrying the stimulus of a square (top-left), triangle (bottom-left), diamond (top-right) and a circle (bottom-right). The central network is the reverse network carrying the attentional template. The right-hand column shows LIP layer on the top row, and the disinhibition layers for V2, V4, and PIT below. Red shows positive activations, blue shows negative activations.

Figure 4.10: Activity of the dynamical network with disinhibition after 350ms after deployment of the attentional template for 'square'. The network on the left (vertically) shows the forward network carrying the stimulus of a square (top-left), triangle (bottom-left), diamond (top-right) and a circle (bottom-right). The central network is the reverse network carrying the attentional template. The right-hand column shows LIP layer on the top row, and the disinhibition layers for V2, V4, and PIT below. Red shows positive activations, blue shows negative activations.

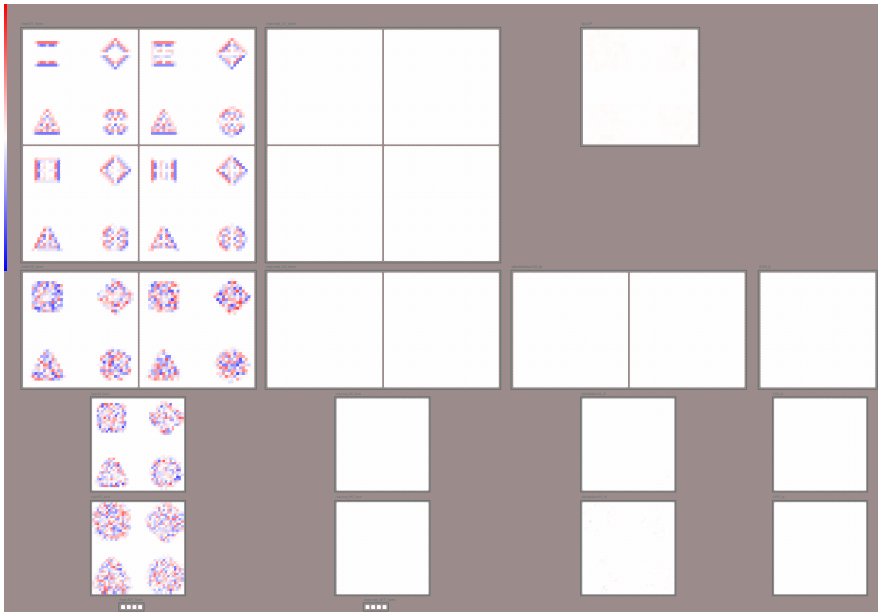


Figure 4.11: Activity of the dynamical network with disinhibition and lateral inhibition after stimulus onset. The network on the left (vertically) shows the forward network carrying the stimulus of a square (top-left), triangle (bottom-left), diamond (top-right) and a circle (bottom-right). The central network is the reverse network carrying the attentional template. The right-hand column shows LIP layer on the top row, and the disinhibition layers for V2, V4, and PIT below. Red shows positive activations, blue shows negative activations.

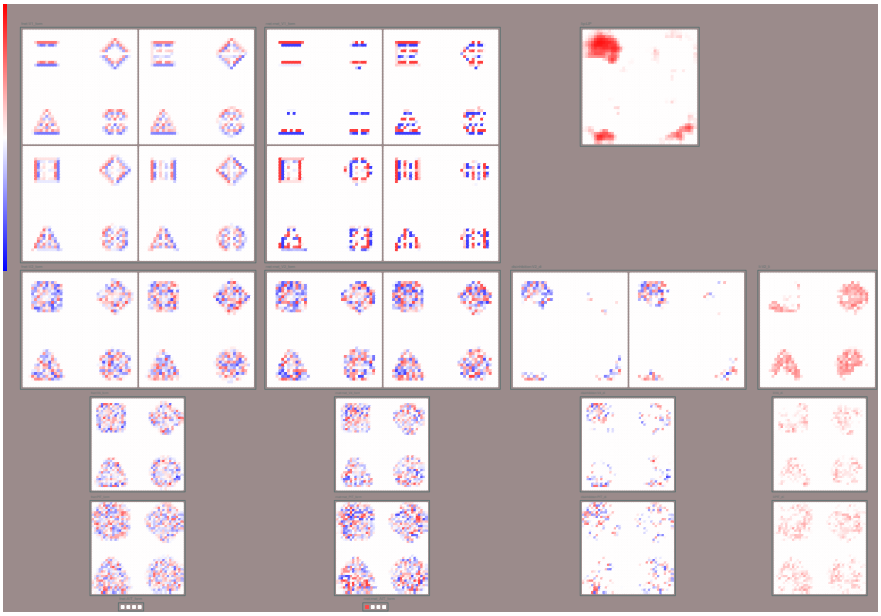


Figure 4.12: Activity of the dynamical network with disinhibition and lateral inhibition after 350ms after deployment of the attentional template for 'square'. The network on the left (vertically) shows the forward network carrying the stimulus of a square (top-left), triangle (bottom-left), diamond (top-right) and a circle (bottom-right). The central network is the reverse network carrying the attentional template. The right-hand column shows LIP layer on the top row, and the disinhibition layers for V2, V4, and PIT below. Red shows positive activations, blue shows negative activations.

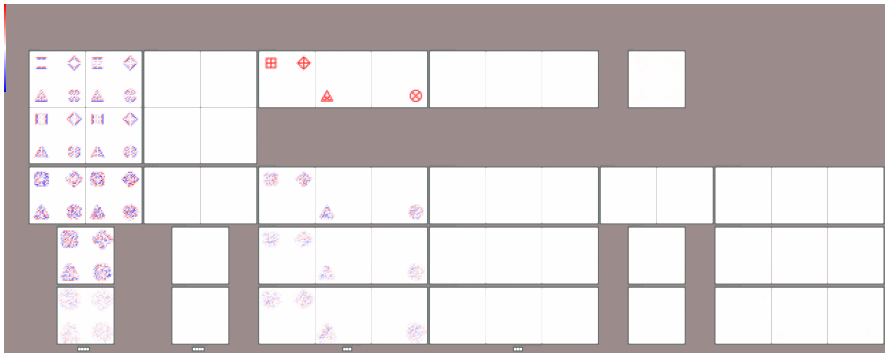


Figure 4.13: Activity of the dynamical network with disinhibition and lateral inhibition and colour channels after stimulus onset. From the left, the networks are the forward form network carrying the stimulus of a red square (top-left), green triangle (bottom-left), red diamond (top-right) and a blue circle (bottom-right); the reverse form network carrying the attentional template; the forward colour network showing red, green and blue channels (left to right), and the reverse colour network; disinhibition layers for V2, V4, and PIT below for form with the LIP layer above; the disinhibition layer for colour, showing red colour channel, green and blue (left to right). Red shows positive activations, blue shows negative activations.

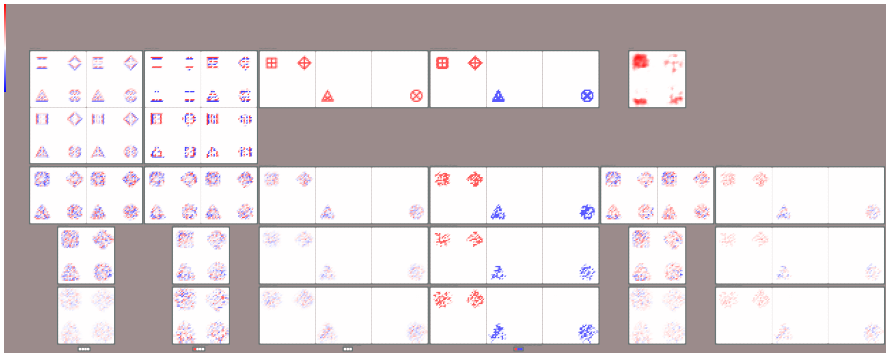
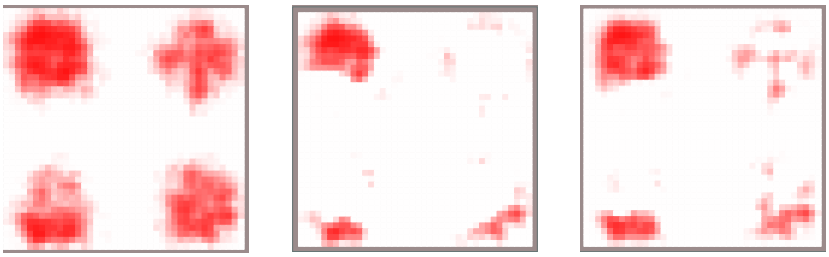


Figure 4.14: Activity of the dynamical network with disinhibition and lateral inhibition and colour channels after 350ms after deployment of the attentional template for 'square' and 'red'. From the left, the networks are the forward form network carrying the stimulus of a red square (top-left), green triangle (bottom-left), red diamond (top-right) and a blue circle (bottom-right); the reverse form network carrying the attentional template; the forward colour network showing red, green and blue channels (left to right), and the reverse colour network; disinhibition layers for V2, V4, and PIT below for form with the LIP layer above; the disinhibition layer for colour, showing red colour channel, green and blue (left to right). Red shows positive activations, blue shows negative activations.

4.4 Discussion of Dynamical Simulation Results



(a) LIP layer, form only, no lateral inhibition.

(b) LIP layer, form only, with lateral inhibition.

(c) LIP layer, colour and form, with lateral inhibition.

Figure 4.15: Close up images of the LIP layer from figures 4.10, 4.12 and 4.14 350ms after application of the attentional template, for each of the dynamical networks.

By observing the activity in the LIP layers in figures 4.15 (a) and (b) it can be seen that 350ms after application of the attentional template there is a larger and stronger area of activity in the top-left quadrant, matching the location of the target object, the square. While the strength of activity in LIP is lower for the the simulation with lateral inhibition (figure 4.15 (b)), the area of activity in location of LIP corresponding to the distractor objects, is smaller than the simulation without lateral inhibition (figure 4.15 (a)). An area based winner-take-all circuit (not implemented) would still select the location of the target object. The fact that the overall LIP activity is lower with lateral inhibition is unsurprising, as the networks are identical except for the addition of extra inhibitory populations.

The results for the simulation with the addition of the colour channels shows a far stronger selection of the target location (4.15 (c), the top-left blob of colour shows the location of the target object), as this simulation has twice the 'evidence' for the location of the target as both colour and form are used to find the target, and their results are combined by the spatially binding effect of LIP. Also, the top-right location corresponds to the location of an input stimulus sharing a feature of the attentional template (the red colour of the diamond object), but its location is inhibited by mismatches in the form feature modality. The process of selection on two features occurs in parallel, so there is no time penalty in using multiple

feature modalities. This is in line with observations from [71, 70]. Furthermore, if it is assumed that the visual system needs to accrue a threshold amount of neural activity in LIP before preparing a saccade to the target, the time to find the target in a visual search task should be less when multiple features are used, as this threshold of 'evidence-based' neural activity would be met sooner.

The use of colour provides a stronger area of neural activity in the ventral stream, as the propagation of form-induced neural activity is edge based, while colour is block based. The ventral stream activity sums to stronger, spatially congruent activity in LIP. Also, the block nature of colour allows the disinhibition network to more effectively block the distractor object, as the locations of mismatched colour as not sparse, as they may be with form processing.

Chapter 5

Conclusion

Contrasting the results of the dynamical models of form with and without lateral inhibition shows that biased competition improves the detection of the target object over a distractor, albeit at a lower activity with lateral inhibition than without. This competition helps to reduce the interference arising in the attentional template caused by presentation of the same object in different locations of a neuron's receptive field. This interference is an inherent consequence of striving for translation invariance, as neurons must learn multiple neural representations of the same object. This is exhibited as a lower degree of matching between the learned attentional template and the stimulus evoked neural trace in the feed-forward network.

As discussed in section 4.2.3, the perceptron circuit can only adopt 2 states when a stimulus is applied to its receptive field. Therefore, a poor match between the learned attentional template and the neural representation of the stimulus will lead to a greater degree of mismatches. A weak lateral inhibition from these mismatched populations can extinguish excitatory activity in areas of poor matching, but only if the number of inhibitory populations in a local neighbourhood is great enough for the sum of all inhibitory activity to exceed the excitatory input to erroneously matched populations. By utilizing independent matching networks, the level of inhibition can be increased, as a location is observed by multiple receptive fields, with each capable of inhibiting local mismatches.

The results of the dynamical model with the two feature modalities of colour and form (section 4.3.3) demonstrates the interaction of multiple matching subnetworks upon the same location of the visual array improves the matching be-

haviour of the network ensemble as a whole. This combination occurs through an additive process of each modality's 'evidence' for the presence of the feature at a given location: activity from matching locations of all engaged attentional feature streams is projected onto a common neural layer in the dorsal stream (LIP). This supports the hypothesis that biased competition occurs independently for each feature modality, and can be combined from separate streams using a connectionist approach.

Furthermore, the results of the simulation of multiple feature modalities (form and colour) demonstrates a resolution to the binding problem arising from a distributed representation of object properties through spatial congruence. The inclusion of multiple feature detectors takes no additional processing time, making use of the highly parallel nature of the visual cortex to detect the presence of target object more robustly.

5.1 Applicability to Computer Science

Visual search is important as it provides a means to study the deployment of visual attention. Visual attention is important as an effective means to drastically reduce the huge amount of information reaching the eye, to a volume of data that can be processed in near real time. Understanding how attention is deployed has potential benefits to computer vision in general, as humans currently process visual input much faster than non-biological algorithmic computer vision techniques. One reason for this disparity may well be that the use of attention allows the slower computing hardware of the brain (in contrast to a computer) to be directed at a much smaller proportion of the visual input. Of course, the human brain is a massively parallel computer of slow processing units versus the serial but incredibly fast processing power of a modern computer, but as computers are moving towards multiple cores (potentially many thousands of cores) the lessons learned from the study of human visual processing may become more directly applicable to conventional computer vision in the near future.

Bibliography

[1] F. A. C. Azevedo, L. R. B. Carvalho, L. T. Grinberg, J. M. Farfel, R. E. L. Ferretti, R. E. P. Leite, W. J. Filho, R. Lent, and S. Herculano-Houzel. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *The Journal of Comparative Neurology*, 513(5):532–541, 2009. 1

[2] A. Baddeley. Working memory: Looking back and looking forward. *Nature Reviews Neuroscience*, 4(10):829–839, October 2003. 1.6

[3] M. Balzer and O. Deussen. Hierarchy Based 3D Visualization of Large Software Structures. In *VIS '04: Proceedings of the conference on Visualization '04*, page 598.4, Washington, DC, USA, 2004. IEEE Computer Society. 3.4.1

[4] T. Binzegger, R. J. Douglas, and K. Martin. Cortical architecture. In M. De Gregorio, V. Di Maio, M. Frucci, and C. Musio, editors, *BVAI*, volume 3704 of *Lecture Notes in Computer Science*, pages 15–28. Springer, 2005. 1.1.5

[5] J. W. Bisley. The neural basis of visual attention. *The Journal of Physiology*, 589(1):49–57, 2011. 1.2.1

[6] R. T. Born and D. C. Bradley. Structure and function ofvisual area MT. *Annual Review of Neuroscience*, 28:157–89, 2005. 1.1.2.4

[7] G. M. Boynton. Attention and visual perception. *Current Opinion in Neurobiology*, 15(4):465–469, 2005. (document), 1.2.2, 1.4, 1.8

[8] G. M. Boynton. A framework for describing the effects of attention on visual responses. *Vision Research*, 49(10):1129–1143, 2009. 1.2, 1.2.2

[9] J. Buhmann, M. Lades, and C. von der Malsburg. Size and distortion invariant object recognition by hierarchical graph matching. In *Proceedings of International Joint Conference on Neural Networks*, pages 411–416, 1990. 1.1.5, 2.3.3.1

[10] L. Chelazzi, E. K. Miller, J. Duncan, and R. Desimone. A neural basis for visual search in inferior temporal cortex. *Nature*, 363:345–347, 1993. 1.9.1, 4, 4.2.3

[11] A. D. F. Clarke, M. J. Chantler, and P. R. Green. Modeling visual search on a rough surface. *Journal of Vision*, 9(4):1–12, 4 2009. 1.6

[12] F. Crick. Function of the thalamic reticular complex: the searchlight hypothesis. *Proceedings of the National Academies of Sciences USA*, 81(14):4586–4590, 1984. 1

[13] J. G. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of the Optical Society of America A*, 2(7):1160–1169, 1985. 2.3.3.1

[14] J. G. Daugman. Complete discrete 2-d gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1169–1179, 1988. 2.3.3.1

[15] M. de Kamps. A simple and stable numerical solution for the population density equation. *Neural Computation*, 15(9):2129–2146, 2003. 2.1

[16] M. de Kamps, V. Baier, J. Drever, M. Dietz, L. Mösenlechner, and F. van der Velde. The state of MIIND. *Neural Networks*, 21(8):1164–1181, 2008. 1.9.1, 2.1, 4

[17] M. de Kamps and F. van der Velde. From artificial neural networks to spiking neuron populations and back again. *Neural Networks*, 14(6-7):941–953, 2001. 1.9.1, 2.1, 4, 4.1, 4.2.1, 4.3

[18] M. de Kamps and F. van der Velde. Using a recurrent network to bind form, color and position into a unified percept. *Neurocomputing*, 38-40:523–528, 2001. 1.9.3, 3.1, 3.1.1, 4.2.4

[19] M. de Kamps and F. van der Velde. Neural blackboard architectures: the realization of compositionality and systematicity in neural networks. *Journal of Neural Engineering*, 3(1):R1–R12, 2006. 1.9.1

[20] G. Deco and E. T. Rolls. A neurodynamical cortical model of visual attention and invariant object recognition. *Vision Research*, 44:621–642, 2004. 1.9.2

[21] R. Desimone and J. Duncan. Neural mechanisms of selective visual attention. *Annual Review Neuroscience*, 18:193–222, 1995. 1.1.3, 1.6, 1.6, 1.9.1

[22] J. Duncan, G. Humphreys, and R. Ward. Competitive brain activity in visual attention. *Current Opinion in Neurobiology*, 7:255–261, 1997. 1.3

[23] J. Duncan and G. W. Humphreys. Visual search and stimulus similarity. *Psychological review*, 96(3):433–458, July 1989. 1.6, 1.9.2

[24] D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1:1–47, 1991. 1.1

[25] D. J. Field. Relation between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America A*, 4(12):2379–2394, 1987. 2.3.3.1

[26] C. D. Gilbert. Microcircuitry of the visual cortex. *Annual Review of Neuroscience*, 6:217–47, 1983. 1.1.5

[27] C. D. Gilbert and T. N. Wiesel. Functional organization of the visual cortex. *Progress in Brain Research*, 58:209–18, 1983. 1.1.5

[28] M. A. Goodale. Visual pathways supporting perception and action in the primate cerebral cortex. *Current Opinion in Neurobiology*, 3(4):578–585, 1993. 1.1.1

[29] M. A. Goodale. Transforming vision into action. *Vision Research*, 51(13):1567–1587, 2011. 1.1.1

[30] M. A. Goodale and A. D. Milner. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15(1):20–25, 1992. 1.1.1

[31] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006. 3.2.2

[32] S. Hochstein and M. Ahissar. View from the top: Hierarchies and reverse hierarchies in the visual system. *Neuron*, 36:791–804, 2002. 1.6

[33] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962. 1.1.2.1, 1.1.5

[34] L. Itti and C. Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40:1489–1506, 2000. 1.2.1, 1.6

[35] C. Koch and S. Ullman. Shifts in selective visual attention. *Human Neurobiology*, 4(4):219–227, 1985. 1.6

[36] T. Kohonen. *Self-organizing maps*, volume 30 of *Information sciences*. Springer, 3 edition, 2001. 3.3

[37] H. Komatsu. Mechanisms of central color vision. *Current Opinion in Neurobiology*, 8(4):503–508, 1998. 3.1.1.1

[38] V. A. F. Lamme and P. R. Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends in Neurosciences*, 23(11):571–579, 2000. 1.1.2.1

[39] A.B. Leber and H.E. Egeth. It's under control: Top-down search strategies can override attentional capture. *Psychonomic Bulletin & Review*, 13(1):132–138, 2006. 1.6

[40] S. P. MacEvoy, T. R. Tucker, and D. Fitzpatrick. A precise form of divisive suppression supports population coding in the primary visual cortex. *Nature Neuroscience*, 12(5):637–645, April 2009. 1.9.1, 4

[41] J. C. Martinez-Trujillo and S. Treue. Feature-based attention increases the selectivity of population responses in primate visual cortex. *Current Biology*, 14(9):744–751, 2004. 1.2.2, 1.4

[42] D. C. Maunsell, J. H. Van Essen. Functional properties of neurons in middle temporal visual area of the macaque monkey. i. selectivity for stimulus direction, speed, and orientation. *Journal of Neurophysiology*, 49(5):1127–47, 5 1983. 1.1.2.4

[43] J. H. Maunsell and W. T. Newsome. Visual processing in monkey extrastriate cortex. *Annual Review of Neuroscience*, 10:363–401, 1987. 1.1.2.4

[44] J. H. R. Maunsell and S. Treue. Feature-based attention in visual cortex. *Trends in Neurosciences*, 29(6):317–322, 2006. 1.2.2, 1.4

[45] C. McAdams and J. H. Maunsell. Effects of attention on orientation-tuning functions of single neurons in macaque cortical area v4. *Journal of Neuroscience*, 19:431–441, 1999. 1.4

[46] M. Mishkin, L. G. Ungerleider, and K. A. Macko. Object vision and spatial vision: two cortical pathways. *Trends in Neurosciences*, 6:414–417, 1983. 1.1.1, 1.9.1

[47] J. Moran and R. Desimone. Selective attention gates visual processing in the extrastriate cortex. *Science*, 229(4715):782–784, 1985. 1.1.2.3, 1.2.3

[48] B. C. Motter. Focal attention produces spatially selective processing in visual cortical areas v1, v2, and v4 in the presence of competing stimuli. *Journal of Neurophysiology*, 70(3):909–119, 1993. 1.2.1

[49] B. C. Motter. Neural correlates of attentive selection for color and luminance in extrastriate area v4. *Journal of Neuroscience*, 14:2178–2189, 1994. 1.4

[50] B. C. Motter. Neural correlates of feature selective memory and pop-out in extrastriate area v4. *The Journal of Neuroscience*, 14(4):2190–2199, 1994. 1.4

[51] V. B. Mountcastle. An organizing principle for cerebral function: The unit model and the distributed system. In G. M. Edelman and V. B. Mountcastle, editors, *The Mindful Brain*, pages 7–50. MIT Press, 1978. 1.1.5

[52] M. M. Müller, S. Andersen, N. J. Trujillo, P. Valdés-Sosa, P. Malinowski, and S. A. Hillyard. Feature-selective attention enhances color signals in early visual areas of the human brain. *Proceedings of the National Academy of Sciences*, 103(38):14250–14254, 2006. (document), 1.8

[53] R. A. Rensink. Seeing, sensing and scrutinizing. *Vision Research*, 40:1469–1487, 2000. 1.2.3

[54] J. H. Reynolds and D. J. Heeger. The normalization model of attention. *Neuron*, 61(2):168–185, 2009. 1.9.1

[55] T. Reynolds, J. H. Pasternak and R. Desimone. Attention increases sensitivity of v4 neurons. *Neuron*, 26(3):703–714, 2000. 1.2.1

[56] D. L. Robinson, E. M. Bowman, and G. B. Stanton. Parietal association cortex in the primate: sensory mechanisms and behavioral modulations. *Journal of Neurophysiology*, 53:910–932, 1978. 1.1.4.1

[57] E. T. Rolls and G. Deco. *The Computational Neuroscience of Vision*. Oxford University Press, 2002. 1.1.2.1, 1.1.2.2

[58] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 10 1986. 3.2.2, 4

[59] M. Sàenz, G. T. Buracas, and G. M. Boynton. Global effects of feature-based attention in human visual cortex. *Nature Neuroscience*, 5:631–632, 2002. 1.8

[60] M. Sàenz, G. T. Buracas, and G. M. Boynton. Global feature-based attention for motion and color. *Vision Research*, 43(6):629–637, 2003. (document), 1.4, 1.8

[61] S. J. Schein and R Desimone. Spectral properties of V4 neurons in the macaque. *Journal of Neuroscience*, 10(10):3369–3389, 1990. 3.1.1.1

[62] M. A. Schoenfeld, C. Tempelmann, A. Martinez, J. M. Hopf, C. Sattler, H. J. Heinze, and S. A. Hillyard. Dynamics of feature binding during object-selective attention. *Proceedings of the National Academy of Sciences*, 100(20):11806–11811, 2003. 1.2.3

[63] M. A. Segraves and M. E. Goldberg. Functional properties of corticotectal neurons in the monkey's frontal eye field. *Journal of Neurophysiology*, 58:1387–1419, 1987. 1.1.4.2

[64] E. Seidemann and W. T. Newsome. Effect of spatial attention on the responses of area mt neurons. *Journal of Neurophysiology*, 81:1783–1794, 1999. 1.4

[65] J. T. Serences and G. M. Boynton. Feature-based attentional modulations in the absence of direct visual stimulation. *Neuron*, 55(2):301–312, 2007. 1.2, 1.2.2

[66] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343. IEEE Computer Society Press, 1996. 3.4.1

[67] D. J. Simons. Attentional capture and inattentional blindness. *Trends in Cognitive Sciences*, 4(4):147–155, 4 2000. 1.5

[68] L. C. Sincich and J. C. Horton. The circuitry of v1 and v2: Integration of color, form, and motion. *Annual Review of Neuroscience*, 28:303–326, 2005. 3.5.1

[69] T. P. Trappenberg. *Fundamentals of Computational Neuroscience*. Oxford University Press, 2002. 4.1

[70] A. Treisman and S. Sato. Conjunction search revisited. *Journal of Experimental Psychology: Human Perception and Performance*, 16(3):459–478, 8 1990. 1.6, 4.4

[71] A. M. Treisman and G. Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12(1):97–136, 1980. 1.2.2, 1.4, 1.6, 1.6, 4.4

[72] S. Treue and J. C. Martinez-Trujillo. Feature-based attention influences motion processing gain in macaque visual cortex. *Nature*, 399:575–579, 1999. 1.2.1, 1.4

[73] J. K. Tsotsos. Analyzing vision at the complexity level. *Brain & Behavioral Sciences*, 13(3):423–445, 1990. 1

[74] L. G. Ungerleider and M. Mishkin. Two cortical visual systems. In D. J. Ingle, M. A. Goodale, and R. J. W Mansfield, editors, *Analysis of Visual Behavior*, pages 549–586. MIT Press, Cambridge, MA, 1982. 1.1.1, 1.1.2.4

[75] F. van der Velde and M. de Kamps. From knowing what to knowing where: Modeling object-based attention with feedback disinhibition of activation. *Journal of Cognitive Neuroscience*, 13(4):479–491, 2001. 1.2.2, 1.9.1, 4.2.3

[76] F. van der Velde and M. de Kamps. Neural blackboard architectures of combinatorial structures in cognition. *Behavioral and Brain Sciences*, 29:37–70, 2006. 3.1, 4.2.4

[77] F. van der Velde and M. de Kamps. A neural model of global visual saliency. In S. Vosniadou, D. Kayser, and A. Protopapas, editors, *Proceedings of the European Cognitive Science Conference*, pages 383–388, New York, 2007. Lawrence Erlbaum. 1.6, 1.9.1

[78] F. van der Velde, M. de Kamps, and G. T. van der Voort van der Kleij. CLAM: Closed-Loop Attention Model for visual search. *Neurocomputing*, 58-60:607–612, 2004. 1.9.1, 3.1.1.2, 3.2.3.1, 3.3

[79] D. Walther, L. Itti, M. Riesenhuber, T. Poggio, and C. Koch. Attentional selection for object recognition: A gentle way. In *BMCV '02: Proceedings of the Second International Workshop on Biologically Motivated Computer Vision*, pages 472–479, London, UK, 2002. Springer-Verlag. (document), 1.8

[80] A. B. Watson and J. A. Solomon. Model of visual contrast gain control and pattern masking. *Journal of the Optical Society of America*, 14(9):2379–2391, 1997. 1.2.1

[81] H. R. Wilson and J. D. Cowan. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical Journal*, 12(1):1–24, 1972. 2.1

[82] H. R. Wilson and J. D. Cowan. A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Biological Cybernetics*, 13:55–80, 1973. 2.1, 4, 4.1

[83] J. M. Wolfe. Guidance of visual search by preattentive information. In Laurent Itti, Geraint Rees, and John K. Tsotsos, editors, *Neurobiology of Attention*, pages 101–104. Academic Press, Burlington, 2005. 1.6

[84] J. M. Wolfe, S. J. Butcher, C. Lee, and M. Hyle. Changing Your Mind: On the Contributions of Top-Down and Bottom-Up Guidance in Visual Search for Feature Singletons. *Journal of Experimental Psychology: Human Perception and Performance*, 29(2):483–502, 2003. 1.6

[85] J. M. Wolfe, K. R. Cave, and S. L. Franzel. Guided search: An alternative to the feature integration model for visual search. *Journal of Experimental Psychology: Human Perception and Performance*, 15(3):419–433, 1989. 1.6, 1.6

[86] J. M. Wolfe and T.S. Horowitz. What attributes guide the deployment of visual attention and how do they do it? *Nature Reviews Neuroscience*, 5(6):495–501, 2004. 1.6

[87] J. M. Wolfe and J. Reynolds. Visual search. In Allan I. Basbaum, Akimichi Kaneko, Gordon M. Shepherd, Gerald Westheimer, Thomas D. Albright, Richard H. Masland, Peter Dallos, Donata Oertel, Stuart Firestein, Gary K. Beauchamp, M. Catherine Bushnell, Jon H. Kaas, and Esther Gardner, editors, *The Senses: A Comprehensive Reference*, pages 275–280. Academic Press, New York, 2008. 1.2.1, 1.6, 1.9.1

[88] S. Yantis. To see is to attend. *Science*, 299(5603):54–56, 2003. 1.2

[89] S. M. Zeki. The functional organization of projections from striate to prestriate visual cortex in the rhesus monkey. *Cold Spring Harbor Symposia on Quantitive Biology*, 40:591–600, 1976. 1.1.2.3

[90] W. W. Zhang and S. J. Luck. Feature-based attention modulates feedforward visual processing. *Nature Neuroscience*, 12(1):24–25, 2009. 1.2.2, 1.8

Appendix A

A.1 Results for No-Opposition

Template	Stimulus	Layer	Diff/Fwd	Total
S	S	V2	0.0447029185	
		V4	0.088589653	
		PIT	0.1835468939	0.3168394654
	D	V2	0.1186126568	
		V4	0.1209828352	
		PIT	0.1979577249	0.4375532168
	O	V2	0.1024939413	
		V4	0.130410896	
		PIT	0.2374141795	0.4703190169
	T	V2	0.1244667159	
		V4	0.2060411806	
		PIT	0.2932683461	0.6237762425

Table A.1: Results of i) for Square template. Forward network with the real component of Gabor filters applied to V1 forward network. The numbers represent the total activity in the difference network divided by the total activity in the forward network.

Template	Stimulus	Layer	Diff/Fwd	Total
D	S	V2	0.1155588227	
		V4	0.192191824	
		PIT	0.2695640014	0.5773146482
	D	V2	0.0319419408	
		V4	0.0908731337	
		PIT	0.1508581333	0.2736732077
	O	V2	0.0879963822	
		V4	0.1532529391	
		PIT	0.2358068365	0.4770561579
	T	V2	0.1329452345	
		V4	0.2314114161	
		PIT	0.2942349409	0.6585915914

Table A.2: Results of i) for Diamond template. Forward network with the real component of Gabor filters applied to V1 forward network. The numbers represent the total activity in the difference network divided by the total activity in the forward network.

Template	Stimulus	Layer	Diff/Fwd	Total
O	S	V2	0.0970650277	
		V4	0.1368158789	
		PIT	0.2217076827	0.4555885892
	D	V2	0.0584158979	
		V4	0.099048181	
		PIT	0.1366951917	0.2941592705
	O	V2	0.0454473734	
		V4	0.078346959	
		PIT	0.1691495671	0.2929438995
	T	V2	0.126654708	
		V4	0.2009494188	
		PIT	0.2640806878	0.5916848145

Table A.3: Results of i) for Circle template. Forward network with the real component of Gabor filters applied to V1 forward network. The numbers represent the total activity in the difference network divided by the total activity in the forward network.

Template	Stimulus	Layer	Diff/Fwd	Total
T	S	V2	0.152910901	
		V4	0.2215226557	
		PIT	0.3061916613	0.6806252181
	D	V2	0.1652449781	
		V4	0.2039918852	
		PIT	0.2823904124	0.6516272757
	O	V2	0.150200195	
		V4	0.2113811667	
		PIT	0.3074547185	0.6690360801
	T	V2	0.04727072	
		V4	0.1044881343	
		PIT	0.2100816483	0.3618405026

Table A.4: Results of i) for Triangle template. Forward network with the real component of Gabor filters applied to V1 forward network. The numbers represent the total activity in the difference network divided by the total activity in the forward network.

A.2 Results for Orthogonal Angles Input in Opposition

Template	Stimulus	Layer	Diff/Fwd	Total
S	S	V2	0.1148962291	
		V4	0.1352048731	
		PIT	0.1399642519	0.390065354
	D	V2	0.2048881775	
		V4	0.2397062557	
		PIT	0.2391446602	0.6837390934
	O	V2	0.2723207094	
		V4	0.243137216	
		PIT	0.2409175135	0.7563754389
	T	V2	0.2865657232	
		V4	0.2821677105	
		PIT	0.2699174126	0.8386508462

Table A.5: Results ii) for Square template. Forward network with the real component of Gabor filters applied to V1 forward network and the real component of Gabor filters from lines orthogonal to each V1 filter applied as negative input. The numbers represent the degree of mismatches between the stimulus and the attentional template.

Template	Stimulus	Layer	Diff/Fwd	Total
D	S	V2	0.237710016	
		V4	0.2860965291	
		PIT	0.2805243364	0.8043308816
	D	V2	0.1046090556	
		V4	0.0930898629	
		PIT	0.1016954223	0.2993943408
	O	V2	0.2340739594	
		V4	0.2127463689	
		PIT	0.211079126	0.6578994544
	T	V2	0.2846291193	
		V4	0.2669070529	
		PIT	0.2512482668	0.8027844389

Table A.6: Results of i) for Diamond template. Forward network with the real component of Gabor filters applied to V1 forward network. The numbers represent the total activity in the difference network divided by the total activity in the forward network.

Template	Stimulus	Layer	Diff/Fwd	Total
O	S	V2	0.2273044826	
		V4	0.2112432252	
		PIT	0.2078553791	0.6464030869
	D	V2	0.23103539	
		V4	0.1915407565	
		PIT	0.1924575678	0.6150337142
	O	V2	0.0923904829	
		V4	0.10696771	
		PIT	0.1073279069	0.3066860998
	T	V2	0.2763950741	
		V4	0.2496982392	
		PIT	0.2346463363	0.7607396497

Table A.7: Results of i) for Circle template. Forward network with the real component of Gabor filters applied to V1 forward network. The numbers represent the total activity in the difference network divided by the total activity in the forward network.

Template	Stimulus	Layer	Diff/Fwd	Total
T	S	V2	0.2724194872	
		V4	0.3859057528	
		PIT	0.3770801022	1.0354053422
	D	V2	0.2525733057	
		V4	0.3732934474	
		PIT	0.3543090019	0.9801757549
	O	V2	0.3040699209	
		V4	0.423244342	
		PIT	0.406980653	1.1342949159
	T	V2	0.0987018712	
		V4	0.1115619782	
		PIT	0.1197312061	0.3299950555

Table A.8: Results of i) for Triangle template. Forward network with the real component of Gabor filters applied to V1 forward network. The numbers represent the total activity in the difference network divided by the total activity in the forward network.

A.3 Results for Real and Imaginary Inputs in Opposition

Template	Stimulus	Layer	Diff/Fwd	Total
S	S	V2	0.1391488765	
		V4	0.2783221196	
		PIT	0.3410666449	0.758537641
	D	V2	0.2578198488	
		V4	0.364239496	
		PIT	0.3414717518	0.9635310965
	O	V2	0.328718476	
		V4	0.4912222935	
		PIT	0.490603361	1.3105441305
	T	V2	0.4066341853	
		V4	0.5326675406	
		PIT	0.5867942192	1.5260959451

Table A.9: Results of iii) for Square template. Forward network with the real component of Gabor filters applied to V1 forward network and the imaginary component applied as negative input. The numbers represent the total activity in the difference network divided by the total activity in the forward network.

Template	Stimulus	Layer	Diff/Fwd	Total
D	S	V2	0.3485902614	
		V4	0.4645730624	
		PIT	0.4169869882	1.2301503119
	D	V2	0.0710264734	
		V4	0.1858715102	
		PIT	0.2604600815	0.5173580651
	O	V2	0.3548840646	
		V4	0.3952283988	
		PIT	0.3368511781	1.0869636415
	T	V2	0.543546789	
		V4	0.5829837109	
		PIT	0.6101884317	1.7367189316

Table A.10: Results of iii) for Diamond template. Forward network with the real component of Gabor filters applied to V1 forward network and the imaginary component applied as negative input. The numbers represent the total activity in the difference network divided by the total activity in the forward network.

Template	Stimulus	Layer	Diff/Fwd	Total
O	S	V2	0.3759263706	
		V4	0.404318849	
		PIT	0.4532924464	1.233537666
	D	V2	0.4331560205	
		V4	0.3964700007	
		PIT	0.3605512311	1.1901772523
	O	V2	0.1260106661	
		V4	0.2040522259	
		PIT	0.2294532939	0.5595161858
	T	V2	0.4623294853	
		V4	0.5890424126	
		PIT	0.5584010876	1.6097729854

Table A.11: Results of iii) for Circle template. Forward network with the real component of Gabor filters applied to V1 forward network and the imaginary component applied as negative input. The numbers represent the total activity in the difference network divided by the total activity in the forward network.

Template	Stimulus	Layer	Diff/Fwd	Total
T	S	V2	0.4755586031	
		V4	0.5101123133	
		PIT	0.4138883765	1.3995592929
	D	V2	0.6078516536	
		V4	0.6491298697	
		PIT	0.4910842215	1.7480657449
	O	V2	0.5156615328	
		V4	0.5678328001	
		PIT	0.5037186382	1.5872129711
	T	V2	0.0523878513	
		V4	0.1493678525	
		PIT	0.1575733356	0.3593290393

Table A.12: Results of iii) for Triangle template. Forward network with the real component of Gabor filters applied to V1 forward network and the imaginary component applied as negative input. The numbers represent the total activity in the difference network divided by the total activity in the forward network.

A.4 Parametrization of Sigmoid Functions for Evaluation in section 4.3

Network	Type	Layer	τ	Max Rate	Power	Noise
ANN	N/A	V1	N/A	1	1	1.5
Dyn	E	V1	$1 \times 10^{-}3$	100	1.0	1.5
Dyn	I	V1	$5 \times 10^{-}4$	100	1.0	1.5
ANN	N/A	V1	N/A	1	1	1.2
Dyn	E	V2	$1 \times 10^{-}3$	100	1.0	1.2
Dyn	I	V2	$5 \times 10^{-}4$	100	1.0	1.2
ANN	N/A	V4	N/A	1	1	1.0
Dyn	E	V4	$1 \times 10^{-}3$	100	1.0	0.2
Dyn	I	V4	$5 \times 10^{-}4$	100	1.0	0.2
ANN	N/A	PIT	N/A	1	1	1.0
Dyn	E	PIT	$1 \times 10^{-}3$	100	1.0	0.2
Dyn	I	PIT	$5 \times 10^{-}4$	100	1.0	0.2
ANN	N/A	AIT	N/A	1	1	1.5
Dyn	E	AIT	$1 \times 10^{-}3$	100	1.0	1.0
Dyn	I	AIT	$5 \times 10^{-}4$	100	1.0	1.0
LIP	E	LIP	$1 \times 10^{-}3$	100	1.0	0.5

Table A.13: Parametrization of the Nodes of the forward networks and LIP network in section 4.3. The dynamical populations of the forward dynamical networks are for *e_p*, *i_p*, *e_n* and *i_n*, as these vary based on the size of the receptive fields. E is excitatory, I is inhibitory.

Network	Type	Layer	τ	Max Rate	Power	Noise
rANN	N/A	All	N/A	1	1	1.5
rDyn	E	All	$1 \times 10^{-}3$	100	1.0	1.0
rDyn	I	All	$5 \times 10^{-}4$	100	1.0	1.0

Table A.14: Parametrization of the Nodes of the reverse networks in section 4.3. The dynamical populations are for *e_p*, *i_p*, *e_n* and *i_n*, as these vary based on the size of the receptive fields. E is excitatory, I is inhibitory.

Name	Type	Layer	τ	Max Rate	Power	Noise
p_out	E	All	$1 \times 10^{-}3$	100	1.0	1.0
n_out	E	All	$5 \times 10^{-}4$	100	1.0	1.0

Table A.15: Parametrization of the Perceptron Circuit output nodes (*p_out* and *n_out*) of the forward and reverse dynamical networks in section 4.3. E is excitatory, I is inhibitory.

Name	Type	τ	Max Rate	Power	Noise
e_dis_p	E	$1 \times 10^{-}3$	100	1.0	1.25
i_dis_p	I	$5 \times 10^{-}4$	100	1.0	1.25
e_rev_p	E	$1 \times 10^{-}3$	100	1.0	1.25
i_gat_p	I	$5 \times 10^{-}4$	100	1.0	1.25
e_dis_n	E	$1 \times 10^{-}3$	100	1.0	1.25
i_dis_n	I	$5 \times 10^{-}4$	100	1.0	1.25
e_rev_n	E	$1 \times 10^{-}3$	100	1.0	1.25
i_gat_n	I	$5 \times 10^{-}4$	100	1.0	1.25
e_r_n	E	$1 \times 10^{-}3$	100	1.0	1.25
i_r_n	I	$5 \times 10^{-}4$	100	1.0	1.25
e_r_p	E	$1 \times 10^{-}3$	100	1.0	1.25
i_r_p	I	$5 \times 10^{-}4$	100	1.0	1.25
e_f_in	E	$1 \times 10^{-}3$	100	1.0	1.25
i_gat	I	$5 \times 10^{-}4$	100	1.0	1.25
e_dis	E	$1 \times 10^{-}3$	100	1.0	1.25
i_dis	I	$5 \times 10^{-}4$	100	1.0	1.25
i_ili	I	$5 \times 10^{-}4$	100	1.0	1.25
i_li	I	$5 \times 10^{-}4$	100	1.0	1.25

Table A.16: Parametrization of the Disinhibition Circuit nodes of the dynamical networks in section 4.3. E is excitatory, I is inhibitory.