An Empirical Model of Area MT: Investigating the Link between Representation Properties and Function

by

Seyed Omid Sadat Rezai

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2018

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner	Duje Tadin Professor, Department of Brain and Cognitive Sciences, University of Rochester
Supervisor	Bryan Tripp Associate Professor, Department of Systems Design Engineering, University of Waterloo
Internal Member	Chris Eliasmith Professor, Department of Systems Design Engineering, University of Waterloo
Internal Member	John Zelek Associate Professor, Department of Systems Design Engineering, University of Waterloo
Internal-External Member	Britt Anderson Associate Professor, Department of Psychology, University of Waterloo

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

I am the sole author of Chapters 1, 2, 3, 6, and 8.

Chapters 4 and 7 are part of an article (Seyed Omid Sadat Rezai, Pinar Boyraz, and Bryan Tripp [172]). All authors participated in writing of the article. The neural data used in Section 4.2.7 was collected by Pinar Boyraz in a previous study [27]. The presented analysis in Section 7.3.2 was conducted by Bryan Tripp and Seyed Omid Sadat Rezai. For the other sections of the article, Seyed Omid Sadat Rezai designed all the models and networks, conducted all the experiments and simulations, and generated all the tables and figures.

Sections 5.2 and 5.3 of Chapter 5 have been adapted from the same article [172].

Abstract

The middle temporal area (MT) is one of the visual areas of the primate brain where neurons have highly specialized representations of motion and binocular disparity. Other stimulus features such as contrast, size, and pattern can also modulate MT activity. Since MT has been studied intensively for decades, there is a rich literature on its response characteristics. Here, I present an empirical model that incorporates some of this literature into a statistical model of population response. Specifically, the parameters of the model are drawn from distributions that I have estimated from data in the electrophysiology literature. The model accepts arbitrary stereo video as input and uses computer-vision methods to calculate dense flow, disparity, and contrast fields. The activity is then predicted using a combination of tuning functions, which have previously been used to describe data in a variety of experiments. The empirical model approximates a number of MT phenomena more closely than other models as well as reproducing three phenomena not addressed with the past models. I present three applications of the model. First, I use it for examining the relationships between MT tuning features and behaviour in an ethologically relevant task. Second, I employ it to study the functional role of MT surrounds in motion-related tasks. Third, I use it to guide the internal activity of a deep convolutional network towards a more physiologically realistic representation.

Acknowledgements

First and foremost, I would like to thank my supervisor, Bryan Tripp. Bryan has been a great mentor during my Master's and PhD. I am grateful for his insightful comments and feedback through all these years.

I would also like to thank the current and past members of the BRAIN lab at UWaterloo for their helpful discussions and comments.

I was lucky to make wonderful friends in Waterloo, of whom some have become a shoulder to lean on. My thanks to them all.

Finally, I could not have come this far without the continued support of my family. Thank you for your unconditional love, care, and patience.

Dedication

To the loving memory of my father, whom I miss every single day.

Table of Contents

List of Acronyms ix

List of Tables xi

List of Figures xii

1 Introduction 1

1.1 Thesis overview 2

2 Area MT and Dorsal Stream: Brain Circuitry for Motion Perception 5

2.1 Dorsal Visual Stream 5

2.1.1 Primary Visual Cortex (V1) 8

2.1.2 V2, V3, and V3A 9

2.1.3 Medial Superior Temporal (MST) Cortex 10

2.1.4 VIP and 7a 10

2.2 Middle Temporal Visual Area (MT) 11

2.2.1 Functional Structure 12

2.2.2 Receptive Fields 12

2.2.3 Characteristics of Responses to Motion 14
2.2.4 Influencing and Represented Variables 15

3 Computer-Vision Algorithms and Deep Neural Networks 17

3.1 Computer-Vision Algorithms 17
3.1.1 Optic Flow 18
3.1.2 Binocular Disparity 19
3.1.3 Algorithm Selection for Flow and Disparity Estimation 21
3.1.4 Lucas-Kanade for Flow Estimation 22
3.1.5 Contrast 27
3.2 Deep Neural Networks 30
3.2.1 Multilayer Perceptrons (MLPs) 31
3.2.2 Typical Architecture 31
3.2.3 Training MLPs with Backpropagation 32
3.2.4 Convolutional Neural Networks (CNNs) 34
3.2.5 LSTM Networks 41
3.2.6 Dropout 44
3.2.7 Batch Normalization 45

4 A Video-Driven Model of Response Statistics in the Primate Middle Temporal Area 47

4.1 Introduction 47
4.2 Methods 48
4.2.1 Structure of the Empirical Model 48

4.2.2 Input Fields . . . 51
4.2.3 Tuning Functions . . . 55
4.2.4 Model Fitting . . . 58
4.2.5 Dynamics of Component and Pattern Selectivity . . . 59
4.2.6 Comparison With Previous Models . . . 62
4.2.7 Prediction of Unseen MT Data . . . 65
4.3 Results . . . 65
4.3.1 Tuning Curve Approximation Examples . . . 65
4.3.2 Dynamics of Pattern and Component Selectivity . . . 68
4.3.3 Parameter Distributions . . . 77
4.3.4 Neural Response Predictions . . . 79
4.4 Discussion . . . 80

5 Sensitivity Analysis of MT Parameters on Visual Odometry Task 82
5.1 Introduction . . . 82
5.1.1 Visual Odometry . . . 83
5.1.2 VO in Primate Brain . . . 84
5.2 Methods . . . 85
5.2.1 A Novel Visual Odometry Dataset . . . 85
5.2.2 Architecture of the CNN . . . 87
5.2.3 Training . . . 88
5.3 Results . . . 88
5.4 Discussion . . . 90
5.4.1 Future Work . . . 91

6 Functional Role of Suppressive Surround of Area MT **92**

6.1 Introduction . . . 92

6.2 Methods . . . 95

6.2.1 Visual Odometry and Hand-Gesture Recognition Datasets . . . 95

6.2.2 Structure of the MT Model . . . 95

6.2.3 Tuning Fields . . . 99

6.2.4 Architecture of the Networks . . . 100

6.2.5 Training . . . 101

6.2.6 Spatial Profiles of Suppression . . . 102

6.2.7 Replacing Task-Optimized Surrounds with MT-Like Surrounds . . . 103

6.2.8 Motion-Opponency Model . . . 103

6.2.9 Surround-Suppression Strength of Following Convolutional Layers . 104

6.3 Results . . . 105

6.4 Discussion . . . 111

6.4.1 Future Work . . . 115

7 Guiding Deep Representations with an Empirical Model of MT **116**

7.1 Introduction . . . 116

7.2 Methods . . . 117

7.2.1 A Novel Visual Odometry Dataset . . . 117

7.2.2 Architecture . . . 118

7.2.3 Training . . . 120

7.3 Results . . . 121

7.3.1 Odometry Performance 121
7.3.2 Speed and Direction Tuning of CNN Units 125
7.4 Discussion 134

8 Conclusion **136**
8.1 Summary of Contributions 136
8.1.1 A Novel Model of MT 136
8.1.2 A Novel Visual Odometry Dataset 137
8.1.3 Sensitivity Analysis of Direction and Speed Tuning on Odometry . 137
8.1.4 Investigating the Role of Surround in Motion-Related Tasks 138
8.1.5 Guiding Representations in Deep Networks 138
8.2 Future Work 139

References **140**

List of Abbreviations

BB Baker & Bair Model

CDS Component Direction Selective

CNN Convolutional Neural Network

CNN-MT CNN Trained with MT Cost

CNN-O CNN Trained with Odometry Cost

CNN-OMT CNN Trained with Odometry & MT Costs

DS-Sup Direction Selective Suppression

ES Excitation Suppression

Exc Excitation

GABA Gamma-Aminobutyric Acid

GPU Graphical Processing Unit

LKNLN Acronym for Empirical Model (Lucas-Kanade Nonlinear-Linear-Nonlinear)

LSTM Long Short-Term Memory

MLP Multi-Layer Perceptron

MO Motion Opponency

MST Medial Superior Temporal Area

MT Middle Temporal Area of Visual Cortex

NG Nishimoto & Gallant Model

NS-Sup Non-direction Selective Suppression

PDS Pattern Direction Selective

PYRLK Pyramidal Lucas and Kanade

ReLU Rectified Linear Unit

V1 Primary Visual Cortex

V2 Secondary Visual Cortex

V3 Visual Area 3

V3A V3 Accessory (A Visual Area Anterior to V3)

V4 Visual Area 4

VIP Ventral Intraparietal Cortex

List of Tables

4.1 Distribution families used for various tuning parameters. 60

4.2 Summary of RMSE comparison between Empirical model and two other models. 66

4.3 Comparison summary between four different forms of pattern selectivity. . 77

5.1 Structure of the CNN used for sensitivity analysis on odometry. 87

6.1 Structure of the CNN used in visual odometry task. 100

6.2 Structure of the CNN used in gesture recognition task. 101

7.1 Structure of the CNN used in visual odometry task. 119

List of Figures

2.1 Illustration of two visual streams 7

2.2 Illustration of aperture problem 9

2.3 Schematic functional architecture of MT with regard to binocular disparity and direction of motion 13

3.1 Illustration of optic flow in both the retina and the image frames 19

3.2 Illustration of binocular disparity 20

3.3 Qualitative differences in the results of different computer vision methods in disparity estimation 23

3.4 Illustration of the pyramidal Lucas-Kanade optic flow method 26

3.5 Illustration of a single-hidden-layer MLP 33

3.6 Illustration of the forward and backward propagations for a hidden neuron 35

3.7 A typical CNN architecture with two feature stages 37

3.8 Sparse connectivity and weight sharing in CNNs 38

3.9 Illustration of a 2×2 pool in a pooling layer. 39

3.10 Long short-term memory unit 43

4.1 Structure of empirical MT model. 52

4.2 Speed tuning curves of four MT neurons. 69
4.3 Effect of contrast on speed tuning. 70
4.4 Attentional modulation of direction tuning. 71
4.5 Response of MT cell to gratings and plaids placed within different regions of RF. 72
4.6 Disparity tuning curves of four neurons. 73
4.7 Examples of direction tuning of two MT cells to monocular and binocular stimuli. 74
4.8 Two examples of size tuning curves. 75
4.9 Pattern selectivity of empirical model. 76
4.10 Examples of parameter distributions. 78
4.11 Explained variance vs. population size of empirical model. 80
4.12 Explained variance vs. scale parameter of speed-tuning-width distribution. 81

5.1 Two example stereo frames from novel odometry dataset. 86
5.2 Task performance comparison with respect to changing direction-tuning-bandwidth distribution. 89
5.3 Task performance comparison with respect to changing speed-tuning-width distribution. 90

6.1 Six examples of the spatial kernels that best fit neural data. 94
6.2 Structure of MT Model with three components. 97
6.3 Validation loss curves for odometry task. 105
6.4 Validation loss and accuracy curves for the gesture recognition task. 106
6.5 Six examples of MT kernels of the odometry networks. 107

6.6 Six examples of MT kernels of the gesture recognition networks. 108
6.7 Comparison between task-optimized and MT-like surround kernels in the odometry networks 109
6.8 Comparison between task-optimized and MT-like surround kernels in the gesture recognition networks 110
6.9 Box-whisker plots of surround-suppression strength in the first convolutional layer after MT. 112
6.10 Box-whisker plots of surround-suppression strength in the second convolutional layer after MT. 113
6.11 Box-whisker plots of surround-suppression strength in the third convolutional layer after MT. 114

7.1 Structure of the CNN. 118
7.2 Validation loss curves for odometry task in two networks. 122
7.3 Scatter plots of actual vs. predicted self-motion velocities of validation set. 123
7.4 Validation loss curves for MT regression. 124
7.5 Direction-tuning curves of example units in CNN-O. 126
7.6 Speed-tuning curves of example units in CNN-O. 127
7.7 Half-height widths of direction-tuning curves. 127
7.8 Direction-tuning curves of example units in CNN-MT. 128
7.9 Speed-tuning curves of example units in CNN-MT. 129
7.10 Direction-tuning curves of example units in CNN-OMT. 130
7.11 Speed-tuning curves of example units in CNN-OMT. 131
7.12 Correlations between tuning curves with dot stimuli and scene stimuli. . . 132
7.13 Correlations between empirical-model tuning curves and CNN tuning curves. 133

Chapter 1

Introduction

The primate middle temporal visual area (MT) is a part of the dorsal stream (or "vision-for-action" pathway [77]). The unique functional and anatomical properties of MT cells have made it an easily identifiable area within the cortex. Studying response characteristics of MT has helped shaping the idea that different visual areas encode highly specialized aspects of visual information. MT has also been an excellent place for evaluating models of population decoding as its response properties are well understood and its principal inputs are known [25]. MT encodes stimulus motion and stereoscopic depth. It has created the opportunity for studying the neural circuits underlying the computations of motion and depth, and also examining the relationships between neural activity and perception [200, 121, 26].

MT has been extensively studied and much is known about its response properties. Also, microstimulation and lesion studies have confirmed its role in smooth-pursuit eye movement [26, 84], and judgement of motion direction and speed [186, 144, 143]. However, microstimulation or lesion studies can only indicate non-specific causal links between MT activity and function, and are unable to reveal the relationship between specific aspects of MT representation and ethologically relevant functions. For example, it is possible that altering specific tuning properties or population statistics would affect the accuracy of

functions such as smooth-pursuit eye movement, self-motion perception, and motion-based segmentation.

I propose an empirical model that can be used for studying such relationships. The proposed model approximates MT responses by covering a wide range of MT response phenomena, more than previous models. Developing an empirical model (as opposed to a mechanistic model) has given me the advantage of approximating MT response statistics with a high level of detail, without requiring a complete understanding of how these responses emerge in the brain.

The proposed model can also be useful for guiding internal representations of deep neural networks to be more aligned with MT representation. More specifically, the model can provide regression targets for intermediate layers of a deep network, driving it to learn a physiologically realistic representation. The same goal can be achieved, with higher physiological validity, if neural data is used as regression targets. However, collecting neural data in quantities large enough for training deep networks is impractical. This model, on the other hand, can inexpensively generate unlimited training data. Another benefit of using this model to guide deep representations is the ability to manipulate tuning properties. This ability may reveal what mechanisms and structures give rise to which tuning properties.

1.1 Thesis overview

Chapter 2, Area MT and Dorsal Stream: Brain Circuitry for Motion Perception, discusses the brain areas of the dorsal visual stream, which are involved in motion perception, as well as their function. The function and properties of MT are explained in more depth as achieving a better understanding of this area is the goal of the thesis.

Chapter 3, Computer-Vision Algorithms and Deep Neural Networks, has two sections. The first section introduces the notions of optic flow, binocular disparity, and contrast. It also briefly reviews a few methods of flow and disparity estimation. Among these methods,

the Lucas-Kanade algorithm and a modern variation of it are discussed in detail. Finally, the proposed method for calculating local band-limited contrast fields is discussed. Both the modern variation of the Lucas-Kanade algorithm and the local contrast method are used in a novel empirical MT model. The second section reviews deep neural networks, particularly convolutional networks and long short-term memory networks. Also, Dropout and Batch Normalization techniques are introduced. These networks and techniques are used in Chapters 5-7.

Chapter 4, A Video-Driven Model of Response Statistics in the Primate Middle Temporal Area, presents a novel empirical model of MT and describes it in detail. Comparison of the model against two recent MT models in fitting neural data is discussed next. Also, the ability of the empirical model to predict unseen neural data and modelling pattern and component selectivity of MT neurons is shown.

Chapter 5, Sensitivity Analysis of MT Parameters on Visual Odometry Task, describes the application of the empirical model for investigating the influence of MT response properties on task performance. Specifically, the effects of modulating two MT tuning features on solving a visual odometry task are investigated. The results suggest that details of MT tuning have a persistent effect on task performance, despite adaptation of the rest of the network around changes in these details. This chapter also introduces a novel visual odometry dataset, which was generated in Unreal Engine 4 and used for the above-mentioned analyses.

Chapter 6, Functional Role of Suppressive Surround of Area MT, explores the role of MT surrounds in solving motion-related tasks using the empirical model. Specificity, the spatial statistics and functional capability of two groups of surround kernels are compared. The kernels in one group have been optimized for a task (i.e., visual odometry or motion-based gesture recognition), while the kernels of the other group have been fit to neural data. Furthermore, the surround strength of three convolutional layers are compared between different deep networks. The results suggest that a fairly large family of MT surround structures can be effective for solving motion-related tasks. Even networks with

no MT surrounds can learn to effectively solve the tasks by introducing stronger suppressive surrounds in their higher-level layers, compensating for the lack of MT surrounds.

Chapter 7, Guiding Deep Representations with an Empirical Model of MT, illustrates how the empirical model can be used for aligning the internal representations of deep convolutional networks (CNNs) more closely with MT. More specifically, this chapter shows how the empirical MT model used to create more realistic direction and speed tuning in an intermediate layer of the proposed deep CNNs.

Chapter 8, Conclusion, summarizes the main contributions and findings of the thesis along with the directions for future work.

Chapter 2

Area MT and Dorsal Stream: Brain Circuitry for Motion Perception

This chapter has two sections. In the first one, I briefly introduce those areas of the dorsal visual stream that constitute the motion-perception circuity of the primate brain, except the area MT. As modelling and studying tuning features of area MT has been the main theme of the thesis, this area demands a more in-depth introduction that I provide in the second part.

2.1 Dorsal Visual Stream

The two-streams hypothesis (proposed by Mishkin et al. [139] and popularized since Goodale and Milner's paper in 1992 [77]) is a widely-accepted influential model of vision processing in the primate brain. Based on this hypothesis, after the visual information is processed within the occipital lobe it follows two main pathways or streams: the ventral stream and the dorsal stream.

The ventral stream (i.e., the "what pathway") is involved in object and scene recognition. On the other hand, the dorsal stream (i.e., the "where/how pathway") plays a vital

role not only in finding where objects are in space but in providing essential information for interacting with those objects. Figure 2.1a depicts these two streams on a lateral view of a human brain.

Motion perception is one of the main roles of the dorsal stream. Figure 2.1b illustrates the anatomical locations of the motion-sensitive areas of this stream on a macaque brain. I will briefly review what each of these brain areas does after defining the important notion of a visual receptive field as well as describing the neuronal tuning curves, which illustrate how the average response of a neuron changes with respect to change in one or multiple stimulus features.

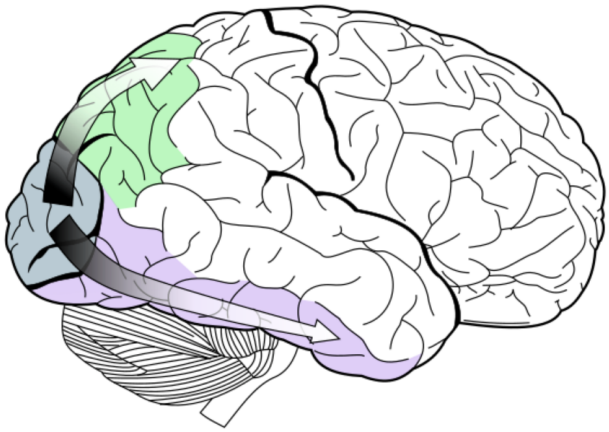
Visual Receptive Field

The receptive field (RF) of a visual neuron comprises a two-dimensional region in the visual space where stimulus presence alters the activity of that neuron. RF of many visual neurons have two subregions: (1) the region where stimulus presence elicits a response whether or not any other stimuli are present a.k.a the classical receptive field (CRF); (2) the region where stimuli cannot elicit a response on its own but can modulate the response of a stimulus in the CRF a.k.a the extra-classical receptive field [86, 167].

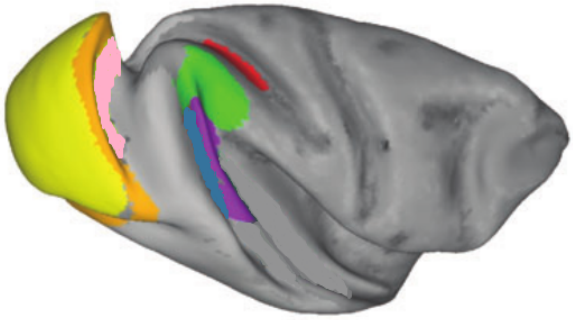
RF sizes increase at successive processing stages (hierarchical levels) in the visual pathway. As well, within the same processing stage (i.e., brain area) a positive correlation exists between RF sizes and the distance of the RF's centres from the point of fixation. Visual RF sizes can range from a few minutes of arc (a dot in this page at reading distance) to tens of degrees (the entire page) [8].

Neuronal Tuning Curves

Neurons selectively represent a particular type of sensory, association, motor, or cognitive information. This selectivity (a.k.a tuning) can be characterized in a plot of the average response (i.e., firing rate) of a neuron as a function of relevant stimulus features. Such a



(a)



(b)

Figure 2.1: (a) Illustration of dorsal (green) and ventral (purple) visual streams on a human brain. (b) Locations of the dorsal-stream areas on an inflated macaque brain, from [32] with modification. These areas are V1 (yellow), V2 (orange), V3 (pink), MT (blue), MST (purple), 7a (green), and VIP (red).

plot is called the tuning curve of that neuron. Neurons in different visual areas are tuned to different properties of the stimulus, which appears in their receptive fields. Often, a single visual neuron is simultaneously tuned to several different features of the stimulus. For example, a V1 neuron is selective to the orientation of patterns (e.g., gratings) and

also their spatial frequency, whereas a typical MT neuron is sensitive to the speed and direction of motion as well as binocular disparity.

2.1.1 Primary Visual Cortex (V1)

Visual information goes from the retina to the lateral geniculate nucleus (LGN), passing through the optic nerve, and then comes directly to the primary visual cortex (V1)[1]. V1 is the lowest-level brain area in the visual cortex hierarchy. The visual cortex itself is a part of the cerebral cortex (outermost layered structure of the brain) responsible for processing visual information.

Area V1 has retinotopic organization, meaning that it contains a complete map of the visual field covered by the two eyes and nearby neurons have RFs that include adjacent portions of the visual field. About fifty percent of the human V1 is devoted to the central two percent of the visual field [219]. V1 neurons are classically divided into two categories based on the structure of their RFs: simple and complex (Hubel and Wiesel [95, 96]).

V1 cells respond strongly to motion of an edge at a certain velocity either depending on (simple cells) or invariant to (complex cells) position within the RF. The majority of V1 cells cannot solve the *aperture problem* [154].

Aperture Problem

If the aperture (receptive field) of a motion detector (visual neuron) is much smaller than the contour it observes, the detector can only be sensitive to the component of the contour's motion perpendicular to the contour's edge, while it will be completely blind to any motion parallel to the contour (apertures 1 and 3 in Figure 2.2). This blindness occurs because movement in parallel direction will not change the appearance of anything within the

[1]Note that in addition to this visual pathway (a.k.a the primary pathway), there is a secondary pathway consists of the superior colliculus of the midbrain.

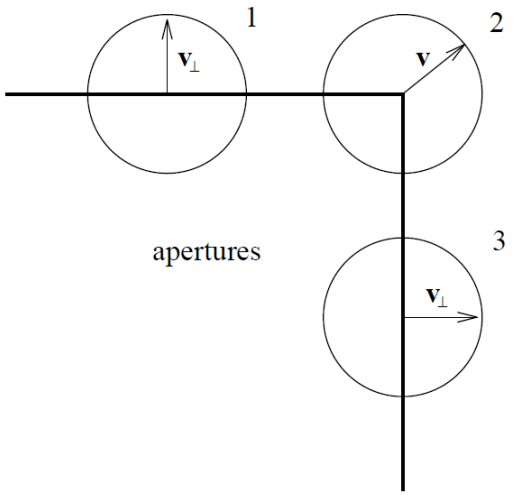


Figure 2.2: Illustration of aperture problem, from [17]. A square is moving up and right. Through apertures 1 and 3 only normal (i.e perpendicular) motions of the edges forming the square can be estimated due to a lack of local structure. From aperture 2, which resides at the corner point, the motion can be fully measured since there is sufficient local structure (i.e., both normal motions are visible) [17].

aperture. As a result, the motion detector is unable to detect the true movement of the contour [88]. Figure 2.2 illustrates the aperture problem.

2.1.2 V2, V3, and V3A

Visual area V2 or the secondary visual cortex, is the second major area in the visual cortex. V2 neurons are retinotopically organized and their major input comes from V1. Some V2 neurons have orientation, colour, and disparity tuning (similar to V1 selectivity) but a fraction of V2 cells are sensitive to relative disparity (as opposed to absolute disparity in V1). These neurons can encode depth relative to another plane rather than absolute depth [112]. The main extra feature of V2 (compared to V1) is the more sophisticated contour

representation, including texture-defined and illusory contours, as well as contours with border ownership [112].

Functional MRI (fMRI) studies suggested that areas V3[2] and V3A[3] may play a role in motion processing [209, 28, 133], however not much is known about V3/V3A specific roles [112]. Most V3 neurons are selective for binocular disparity [2] and project to V3A, which is also strongly activated during binocular disparity processing [215]. Both V3 and V3A project to caudal intraparietal area (CIP) where neurons are selective for depth gradients and curvature [104].

2.1.3 Medial Superior Temporal (MST) Cortex

Medial Superior Temporal (MST) Cortex is the most studied area in the context of self-motion [32]. MST is divided into two main parts: the dorsal (MSTd) and lateral (MSTl). MSTd is more responsive to large-field visual stimuli encoding heading [182]. MST RFs are larger than both V1 and MT RFs but they lack retinotopic organization. MST neurons respond selectively to optical flow components such as expansion, contraction, and clockwise or counterclockwise rotation [184, 183, 182, 64, 153, 114].

While MST is not retinotopically organized, heading is encoded in retinal coordinates (e.g., left or right with respect to the direction of gaze) [112]. MST also receives extraretinal eye-movement information that helps in accurate heading estimation even during eye movements [145, 32].

2.1.4 VIP and 7a

Visual neurons in ventral intraparietal (VIP) respond to complex motion stimuli, such as the direction of heading in optic flow [46]. VIP RFs are independent of gaze direction (in

[2]Third visual area

[3]V3 Accessory, a visual area anterior to V3

contrast to the majority of MST neurons) suggesting that they encode motion information in a head-centric frame [30]. In addition to encoding self-motion, VIP is invloved in control of head movements, and the encoding of near-extrapersonal (head centred) space [46]. VIP sends projections to the polysensory neurons of motor cortex, which are clustered in area F4 [79]. VIP neurons often respond to touch, with RFs around the head and shoulders, aligned with the visual RFs [47]. Also, microstimulation of VIP neurons elicits movements that seem defensive [47]. Areas MST and VIP are cortical substrates for heading perception [32].

Area 7a receives extensive projections from area MST and it contains cells that are narrowly tuned to radial flow and show gain modulation by eye position [191]. Therefore, 7a potentially solves the rotation problem converting an oculocentric (i.e., based on retinal coordinates) heading estimate into a head-centric frame [222].

2.2 Middle Temporal Visual Area (MT)

The middle temporal visual area (MT or V5), discovered at about the same time by Dubner and Zeki [63] in Old World macaque monkeys and by Allman and Kaas [7] in New World owl monkeys, is a well-studied motion-sensitive area in the dorsal stream. Though part of the *extrastriate cortex*[4], MT is still quite close to the retina since its principal inputs are as few as five synapses away from the photoreceptors. This attribute facilitates the characterization of the mechanisms by which the properties of MT receptive fields (RFs) arise [25]. Another important attribute of area MT is that its cells are close enough to some outputs (in particular, those involved in eye movements) to provide an easily measurable continuous readout of computations performed in the pathways which MT is a part of [119]. Together, these attributes have made MT an attractive target of extensive research.

MT sends a strong projection to MST, where cells have larger RFs and encode ego-

[4]The region of the occipital cortex of the mammalian brain located next to the primary visual cortex (i.e., striate cortex), which comprises areas V3, V4, and V5/MT.

motion. Microstimulation studies confirmed the role of MT cells in motion perception where microstimulation biased the animals' judgements towards the direction of motion encoded by the stimulated neurons [186, 187]. Furthermore, lesion studies in monkeys have confirmed the role of MT in smooth pursuit eye movements [119].

MT constitutes a border area between the parietal and occipital lobes (considered to be a part of the latter). The anatomical and histological properties (e.g., being buried in the superior temporal sulcus, receiving direct inputs from V1, and heavy myelination) as well as functional properties of MT cells (e.g., highly responsive to motion and topographically organized RFs) help in determining its borders within the cortex [132].

2.2.1 Functional Structure

MT has a retinotopic organization where each hemisphere contains a somewhat complete map of the contralateral visual hemifield, with a marked emphasis on the fovea. More specifically, the central 15° of the visual field occupies over half of MT's surface area [217] with a bias toward the inferior (i.e., lower) quadrant of the visual field [132].

Within this relatively crude retinotopic map, several other organizations (i.e., visual maps) exist at finer spatial scales, which correspond to neural tuning for different stimulus parameters. Namely, a columnar organization composed of columns of smoothly varying preferred directions running side by side (although some columns occasionally prefer the locally opposite direction) [4], a columnar organization of tuning for binocular disparity (coexisting with the direction columns; see Figure 2.3) [55], and also clusters of neurons by speed preference that are not strictly organized in columns [120].

2.2.2 Receptive Fields

MT cells respond best when stimuli cover the centre of their RFs [167]. In the literature, this central part is referred to as the classical receptive field (CRF) or the excitatory centre

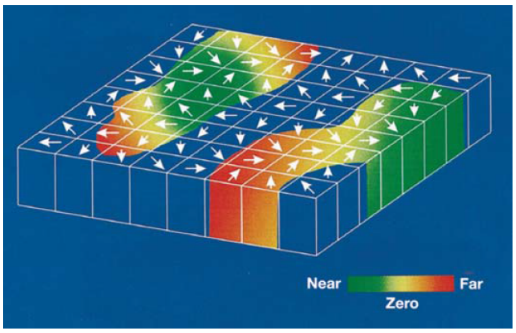


Figure 2.3: Schematic summary of the functional architecture of MT with regard to binocular disparity and direction of motion, from [55]. The top surface of this slab corresponds to the surface of MT, and the height of the slab corresponds to the thickness of the cortex. Arrows denote the preferred direction of motion of MT neurons in each direction column. Direction has been shown to vary smoothly across the surface of MT in both dimensions (no discontinuities in the direction have been depicted). Preferred disparity is colour-coded, with green representing near disparities, red representing far disparities, and yellow indicating zero disparity. Dark blue regions denote portions of MT that have poor disparity tuning [55].

where presence of stimuli excites the cell (i.e., increases its activity). The CRF is generally elongated with the axis of elongation orthogonal to the preferred direction of motion [167], and it is enclosed by an inhibitory surround structure (i.e., the extra-classical receptive field), which can extend for 7-10 times the size of the centre [167]. While the classical view suggested a symmetrical shape for the surround with the same preference for direction of motion [206], as the CRF preference, more recent studies have found more heterogeneous spatial profile for the surround as well as different direction preferences from those of the centre [223]. The spatial heterogeneity of the surround is thought to boost MT's capacity for estimating 3D velocity [49]. This boost can facilitate computation of structure from motion [25] as well as heading perception in MST [49]

2.2.3 Characteristics of Responses to Motion

Almost all MT cells are responsive to the direction and speed of moving stimuli. Some MT cells respond selectively to pattern motion. These cells are capable of coding the motion of whole visual patterns (independent of the motions of contours within their RF), and therefore solve the aperture problem. Some other cells represent the motion of stimulus components, and like V1 cells, are unable to solve the aperture problem [180].

Local Motion Integration

If motion integration were global in MT cells, overlapping and non-overlapping components of a moving pattern would evoke similar cell activity as long as the components resided inside the cell's RF in the second case. However, experiments showed that two different stimuli, one composed of two non-overlapping drifting gratings, and the other composed of two overlapping drifting gratings (which create a plaid pattern) evoke different responses in the pattern-selective cells. Hence, motion integration occurs locally within sub-regions of the RF as oppose to globally across the entire RF [125].

Temporal Properties and Pattern Selectivity

Area MT has rapid dynamics. Its minimum latency is about 30 to 35ms and the median latency is approximately 90ms [87, 168, 31]. Also MT cells respond to quite high temporal frequencies. Typically, they peak in the 3-10Hz range, and most will have cut off by 30-50Hz [31]. Some MT cells also exhibit a dynamic solution to the aperture problem. They initially respond primarily to the component of motion perpendicular to a contour's orientation, but over a period of approximately 60ms the responses gradually shift to encode the true stimulus direction, regardless of orientation [154]. Therefore, the population motion response of MT is dominated by component motion signals but gradually shifts to represent pattern motion [197].

Selectivity for Spatial Frequency

In many MT neurons, the preferred speed depends on spatial frequency of stimuli when exposed to sine-wave gratings [164]. However, if the stimuli are changed to plaids (i.e., superimposition of two sine-wave gratings) the preferred speed dependency to spatial frequency decreases [164]. In case of exposure to square-wave gratings (i.e., superimposition of many sine-wave gratings), MT neurons' preferred speeds become independent of spatial frequency. Consequently, it seems RFs of MT neurons have been developed so that in natural scenes, where there are corners and edges (composed of many different spatial frequencies), they respond to speed independently of spatial frequencies [164].

2.2.4 Influencing and Represented Variables

While nearly all MT neurons are tuned for the direction [130, 55] and speed [130, 164, 149] of visual stimuli, some of them are also selective for binocular disparity [131, 55]. Additionally, stimulus features such as size [167, 156], contrast [156], colour [189], temporal and spatial frequency [164] can evoke or suppress MT activity. Spatial and feature-based attention

also modulate the response in MT cells [210, 189] such that the gain of the tuning curve increases MT but the tuning width does not change [211].

Chapter 3

Computer-Vision Algorithms and Deep Neural Networks

I explain the computer vision tools that I used throughout my thesis in this chapter. For readability, I have divided this chapter into two sections. In the first section, all the non-deep-learning computer-vision notions are explained while the second one explores deep learning topics.

3.1 Computer-Vision Algorithms

The empirical MT model, which I explain in detail in the next chapter, receives optic flow, binocular disparity, and contrast fields as input. This section gives the descriptions for these fields and a comparison between several different algorithms for their estimation. Finally, I finish this section with a detailed overview of the pyramidal Lucas-Kanade method, which is used in the thesis.

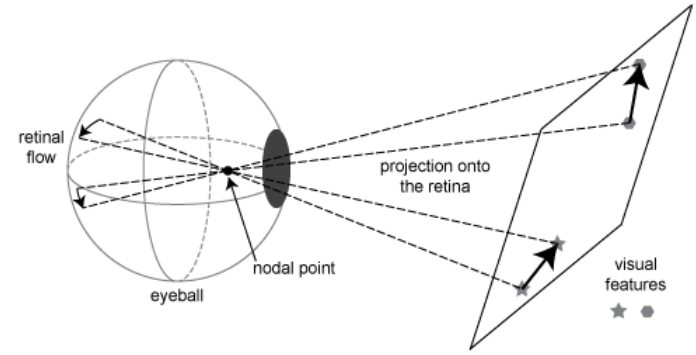
3.1.1 Optic Flow

In the biological context, optic flow (sometimes called retinal velocity) is the change of structured patterns of light on the retina that leads to an impression of movement of the scene projected onto the retina [169].

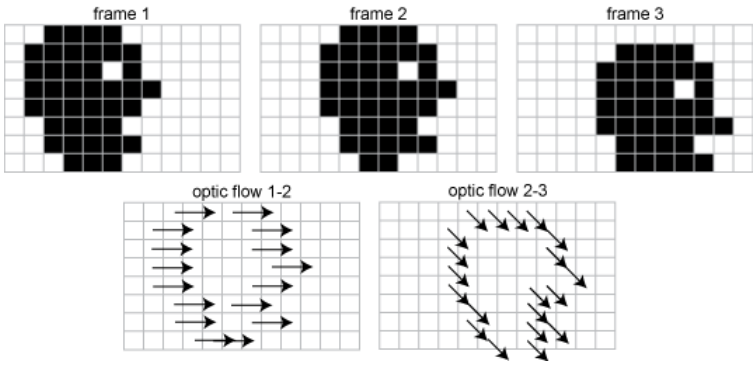
In computer vision applications, the camera becomes the surrogate eye and changes in the environment are represented by a series of image frames. These frames are obtained by a spatiotemporal sampling of the incoming light that hits the camera's sensor. In this context, optic flow is defined as the vector field that captures the displacement of the corresponding pixels in successive frames [17].

Figure 3.1 illustrates the optic flow in both the retina and the image frames. Figure 3.1a shows the movements of two visual features (i.e., star and hexagon) in the environment and their respective optic flow generated on the retina. Figure 3.1b demonstrates three frames illustrating the movement of a head silhouette. The resultant optic flow is depicted as the correspondence between the pixels that represent the contour of the silhouette in consecutive frames.

The first algorithms for the optic flow estimation were proposed in the early eighties [93, 123]. Since then, optic flow has found a variety of applications. Object segmentation and tracking [57], video stabilization [158], video compression [231], and depth estimation [190] are some examples. This wide range of applications has motivated many new algorithms for real-time, pixel-wise (i.e., dense) estimation of optic flow. RNLOD-Flow [230] and Correlation Flow [62] are just some examples of recently proposed optic flow algorithms. Whereas FlowNet [72] and CNN-flow [207] are examples of deep-neural-network approaches. The Middlebury database (`http://vision.middlebury.edu/flow/eval/`) has a benchmark ranking more than 150 different optic flow methods.



(a)



(b)

Figure 3.1: Illustration of optic flow in both the retina and the image frames, from [169].

3.1.2 Binocular Disparity

Binocular disparity is the difference in coordinates of similar features within two stereo frames and can be used to calculate depth of objects within the visual field. Figure 3.2 depicts how images of two different objects, with different depths from a camera system, create different disparities. More specifically, for a pair of calibrated cameras with focal length f that are B unit of distance apart, depth Z of an object can be found from its corresponding disparity d as

$$Z = \frac{fB}{d}, \tag{3.1}$$

where Z and B are often measured in meters while d and f are measured in pixels.

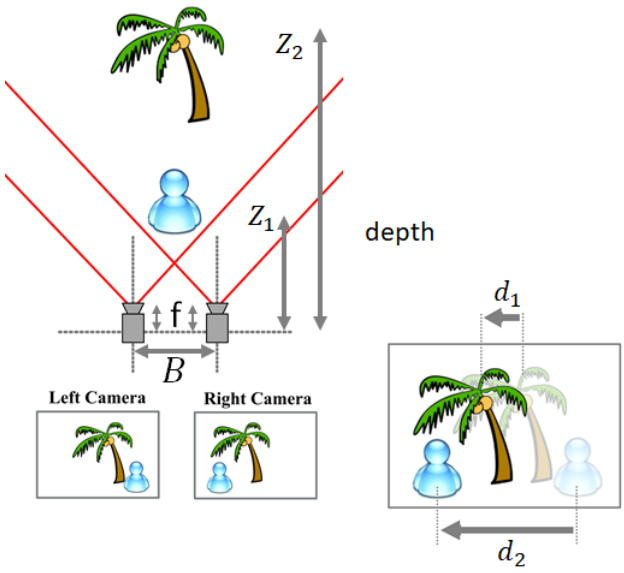


Figure 3.2: Illustration of binocular disparity. The blue figure, which is closer to the camera system, creates a larger disparity compared to the tree, which is more distant.

Disparity estimation can be considered as a special case of optic flow estimation where only horizontal direction is present (zero vertical offset), and inputs are stereo frames as opposed to consecutive frames captured by a single camera. Consequently, almost any optic flow method can potentially be used for disparity estimation as well.

3.1.3 Algorithm Selection for Flow and Disparity Estimation

There is a vast number of algorithms for optic flow or disparity estimation. For accurate estimation, these algorithms should tackle challenges such as occlusion, brightness inconsistency, and the aperture problem (see Section 2.1.1). The earliest optic flow algorithms were proposed in early 1980s. Horn-Schunck [93] and Lucas-Kanade [123] are the best-known examples of these classical methods. They are often computationally cheap but cannot correctly estimate large displacements. However, their modern variations address large displacements by using multi-resolution representations of the original frames (see Section 3.1.4). Another class of algorithms combine the classical formulations with modern optimization and implementation techniques (e.g., Classic++ [201]) to achieve higher performance. Deep-network solutions have been also suggested for calculating both disparity [228] and optic flow [72]. While deep-network methods usually give accurate estimations, they require considerable amount of memory and are slow to run.

Figure 3.3 illustrates a qualitative comparison between eight different methods in disparity estimation. The input was a pair of random-dot images plotted at slightly different horizontal positions. While both classical Lucas-Kanade and Horn-Schunck algorithms failed to properly estimate the ground truth (suffering from aperture problem), some methods extrapolated far beyond well-textured regions, e.g., reporting motion over almost the whole image in response to a small stimulus. This can be interpreted as being physiologically unrealistic because such extrapolations involve lateral communication over the whole visual field.

From Figure 3.3, one can see that the pyramidal Lucas-Kanade neither suffers from aperture problem nor extrapolates beyond the random-dot patches. Also, a parallel im-

plementation of this method can reach 60 FPS, which is one or two orders of magnitude faster compared to some of the above mentioned algorithms. These properties made the pyramidal Lucas-Kanade a good candidate for disparity and optic flow estimation, which were needed for the empirical MT model (see Chapter 4).

3.1.4 Lucas-Kanade for Flow Estimation

The pyramidal Lucas-Kanade [129] is a relatively simple yet accurate flow method. I previously developed a GPU implementation that runs in real time [181]. I used this implementation for calculating flow and disparity fields (see Chapter 4). Here, I explain this algorithm in detail starting first with the classical version.

Classical Lucas-Kanade Algorithm

If the sampling time between the frames is small enough (i.e., high frame rate), we can reasonably assume that the intensity of a visual feature remains approximately constant as it moves from one frame to the next,

$$I(x, y, t) = I(x + dx, y + dy, t + dt). \tag{3.2}$$

By using first order Taylor approximation Equation 3.2 becomes:

$$I(x, y, t) = I(x, y, t) + \frac{\delta I}{\delta x}dx + \frac{\delta I}{\delta y}dy + \frac{\delta I}{\delta t}dt. \tag{3.3}$$

The first term on the right hand side of Equation 3.3 cancels the term on the left hand side,

$$I_x dx + I_y dy + I_t dt = 0, \tag{3.4}$$

where $I_x = \frac{\delta I}{\delta x}$, $I_y = \frac{\delta I}{\delta y}$, and $I_t = \frac{\delta I}{\delta t}$.

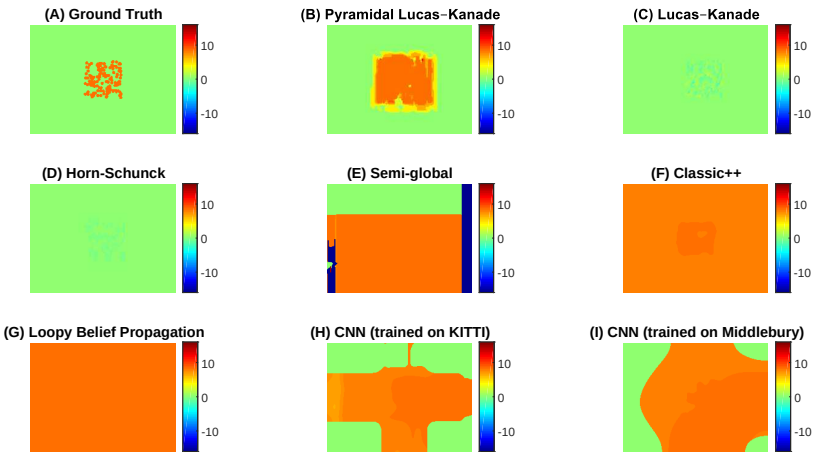


Figure 3.3: Qualitative differences in the results of different computer vision methods in disparity estimation. The input was a pair of random-dot images plotted at slightly different horizontal positions. A, ground truth disparity (the horizontal distance wherever there was a dot; zero elsewhere), B, pyramidal Lucas-Kanade [129], C, classical Lucas-Kanade [123], D, classical Horn-Schunck [93], E, Semi-Global Block Matching [90], F, Classic++ [201] G, loopy belief propagation [69] (running this method with a zero-disparity prior obtained a result that was localized to the stimulus), and H-I, the convolutional neural networks (CNNs) by Žbontar and LeCun [228] trained for disparity estimation on KITTI dataset and Middlebury dataset, respectively. Some of these methods can also be applied to flow, which poses essentially the same matching problem except that the search for each pixel's match is not constrained to the same row of pixels.

Dividing both sides of Equation 3.4 by dt and assuming $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$, gives us the optic flow constraint equation:

$$I_x u + I_y v + I_t = 0, \tag{3.5}$$

where u and v are, respectively, the horizontal and vertical components of the optic flow vector associated with the considered pixel. Note that Equation 3.5 has two unknowns and cannot be solved uniquely. This ambiguity in fact is a demonstration of the aperture problem discussed in Section 2.1.1.

To solve this ambiguity, Lucas and Kanade [123] assumed that the flow is essentially constant in a patch of n pixels centred on the pixel under consideration,

$$\begin{aligned} I_{x_1} u + I_{y_1} v &= -I_{t_1} \\ I_{x_2} u + I_{y_2} v &= -I_{t_2} \\ &\vdots \\ I_{x_n} u + I_{y_n} v &= -I_{t_n}. \end{aligned} \tag{3.6}$$

Writing Equation 3.6 in the matrix form gives:

$$A \begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{b}, \tag{3.7}$$

where:

$$A = \begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \vdots & \vdots \\ I_{x_n} & I_{y_n} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} -I_{t_1} \\ -I_{t_2} \\ \vdots \\ -I_{t_n}. \end{bmatrix} \tag{3.8}$$

To solve Equation 3.7 for u and v, Lucas and Kanade [123] used least squares method that leads to:

$$\begin{bmatrix} u \\ v \end{bmatrix} = (A^T A)^{-1} A^T \mathbf{b}. \tag{3.9}$$

Since $A^T A$ can be a singular matrix, using a regularized least-square method will improve the robustness of the solution [129]. This yields:

$$\begin{bmatrix} u \\ v \end{bmatrix} = (A^T A + \alpha I)^{-1} A^T \mathbf{b}, \tag{3.10}$$

where $0 < \alpha < 10^{-3}$ and I is the identity matrix [129].

The computations related to the classical Lucas-Kanade method stop at this point. One shortcoming of the classic version is its failure in capturing large displacements. To tackle this issue, a pyramid of image frames with coarser resolutions can be built so that large displacements are detected in the coarser copies.

Pyramidal Lucas-Kanade Algorithm

The pyramidal Lucas-Kanade algorithm [129] to estimate optic flow between two frames can be described as follows (see Figure 3.4 for illustration):

1. Building a Gaussian pyramid with n levels:
   (a) The level 0 of the pyramid is filled with the original image.
   (b) For levels $i = 1$ to $n - 1$:
   The level i is built with the image of level $i - 1$ subsampled by a factor of two.
2. Optic flow calculations:
   (a) The optic flow is computed at level $n - 1$ (i.e., the lowest resolution) based on Equation 3.10.

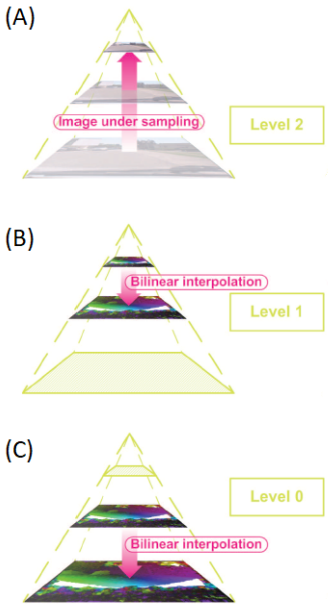


Figure 3.4: Illustration of the pyramidal Lucas-Kanade optic flow method with 3 levels, from [129]. A, flow calculation at level 2: a three-level pyramid is created by subsampling the images of the lower levels. After the pyramid is built, the optic flow is calculated at the highest level (lowest resolution). B, flow calculation at level 1: first the calculated flow at level 2 is oversampled by bilinear interpolation. The resultant flow is used to warp the second frame. Final optic flow at level 1 is the summation of the oversampled flow, multiplied by two, and the flow calculated after warping the second frame. C, flow calculation at level 0 (original resolution): similar to B, first the calculated flow at level 1 is oversampled by bilinear interpolation. The calculated flow is then used to warp the second frame. The final flow is the summation of the oversampled flow, multiplied by two, and the flow calculated between the first frame (at the original resolution) and the warped second frame.

(b) For levels $i = n - 2$ to 0:

i. The initial value is two times the over-sampled optic flow computed at level $i - 1$ using bilinear interpolation.

ii. This initial value is used to warp the second frame.

iii. The optic flow between the first frame and this warped version of the second frame is computed based on Equation 3.10.

iv. The final flow value is the summation of the initial value (Step i) and the value calculated in Step iii.

The number of levels in the pyramid is selected with respect to the original frame resolution and expected optic flow magnitude (typically 3 to 5 levels). Because pixel computations in each level are independent, a parallel implementation on the GPU decreases run time by a few orders of magnitude (compared to the CPU implementation).

3.1.5 Contrast

Contrast is the difference in luminance or colour that makes different visual features distinguishable. The light adaptation process in the retina effectively reduces sensitivity to absolute illumination, which can change several orders of magnitude during the day but is not useful for guiding behaviour [136]. More specifically, the ganglion cells of the retina possess receptive fields with centre and surround regions that are mutually inhibitory [136]. Such receptive field structure makes these cells to be most sensitive to borders and contours (to differences in luminance) as opposed to uniform surfaces [76]. Consequently, the primate visual system is more sensitive to contrast than absolute luminance. Contrast has several different definitions, which we see next.

Weber Contrast

Weber contrast is defined as,

$$c = \frac{I - I_b}{I_b}, \tag{3.11}$$

where I and I_b are, respectively, the luminance of the features and the background. This definition is only useful in cases where small features are present on a uniform background so that the background luminance is a good approximation of the average luminance.

Michelson Contrast

Michelson contrast is defined as

$$c = \frac{I_{max} - I_{min}}{I_{max} + I_{min}}, \tag{3.12}$$

where I_{min} and I_{max} are the lowest and highest luminance. This definition is commonly used for patterns where both bright and dark features are spatially equivalent and take up similar fractions of the area like sinusoidal gratings.

RMS Contrast

Root mean square (RMS) contrast is defined as the standard deviation of the pixel intensities,

$$c = \sqrt{\frac{1}{MN} \sum_{i=1}^{N} \sum_{j=1}^{M} (I_{ij} - \bar{I})^2}, \tag{3.13}$$

where intensity I_{ij} is the i-th row j-th column element of the image of size MxN and $\bar{I}$ is the average intensity of all pixel values in the image.

Peli Contrast

Peli contrast is a local [160], band-limited measure, unlike the previous definitions that are global and frequency-independent. For natural images this definition is the most explanatory as they consist of complex patterns with rich frequency content. Also the empirical MT model needs a local definition to modulate neural responses according to contrast within the receptive fields (as opposed to remote parts of the image). Furthermore, frequency dependence provides a way to match the contrast definition to primate contrast sensitivity [173, 53].

More formally, contrast of a pixel, specified by (x,y), at each spatial frequency band i is defined as a ratio of two functions,

$$c_i(x, y) = \frac{\alpha_i(x, y)}{l_i(x, y)}. \tag{3.14}$$

The numerator function is,

$$\alpha_i(x, y) = I(x, y) * g_i(x, y), \tag{3.15}$$

where I is the image, g_i is a spatial frequency dependent filter, and $*$ denotes convolution. The denominator function is,

$$l_i(x, y) = \bar{I} + \sum_{j=1}^{i-1} \alpha_j(x, y), \tag{3.16}$$

where $\bar{I}$ is the image mean and α_js are the numerator functions corresponding to smaller frequency. Peli [160] suggested cosine log filters as the choice for g_is since an image filtered by a bank of these filters can be reconstructed by a simple addition process without distortion.

For calculating contrast input of the MT model, I used a modified version of Peli's definition. Namely, to relate the contrast definition more directly to the primary visual cortex, I used a bank of Gabor filters (instead of cosine log filters) with four different

frequencies and four different orientations for a total of 16 contrast channels. I combined these channels in a weighted sum:

$$c'(x,y) = \sum_{k=1}^{16} A_k c_k(x,y), \tag{3.17}$$

where $A_k s$ were chosen to approximate macaque contrast sensitivity [53, 54].

I then smoothed the resulting contrast field with a 2D Gaussian kernel, which was meant to approximate integration over V1 cells, and scaled it so that its mean over the image was equal to RMS measure of contrast:

$$c(x,y) = A_{scale} gaussfilt(c'(x,y)). \tag{3.18}$$

The scaling made sure that the global average of the calculated contrast is the same as the RMS contrast, which is widely used in neuroscience literature especially for random dot stimuli [141, 128, 156].

3.2 Deep Neural Networks

Artificial neural networks, inspired by biological neural networks, are composed of interconnected processing elements (neurones) structured to form three kinds of layers: the input layer, the hidden layer(s), and the output layer. A network with one or two hidden layers is called a *shallow network*, whereas a network with more hidden layers is called a *deep network* [20]. Although, universal approximation theorem[1] tells us that theoretically even a shallow network can achieve the excellent problem-solving capabilities of the neural networks, deep networks have been demonstrated to be more effective solutions in practice.

[1] Universal approximation theorem states that a feedforward network with a single or multiple hidden layer(s) containing a finite number of neurons, can approximate continuous functions on compact subsets of $\mathbb{R}^n$, under mild assumptions on the activation function [48].

The power of deep networks stems from the fact that (like the brain) they are capable of learning a hierarchy of features where features in higher levels are formed by the composition of features at lower levels. This means deep networks can learn multiple levels of representations that correspond to different levels of abstraction. In other words, deep networks can automatically and efficiently learn the relevant features, which are essential for solving a task, directly from data as opposed to systems that require human-crafted features as input [21].

In this section, I first introduce multilayer perceptrons (the most basic network architecture). I then explain the convolutional neural networks (CNN; the most common variation of the deep feedforward networks for analyzing visual imagery). I also discuss the long short-term memory (LSTM) networks (a variation of deep recurrent networks, well-suited for making predictions on time series data). Finally, I introduce Dropout and Batch Normalization (effective techniques for improving the performance of deep networks). As later chapters will demonstrate, these networks and techniques have been used in the thesis.

3.2.1 Multilayer Perceptrons (MLPs)

Multilayer perceptrons (MLPs) are feedforward artificial neural networks that, after training, approximate an implicit function that is generalized from input-output examples. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (i.e., processing element) with a nonlinear activation function. Common approaches for training MLPs involve a technique called *backpropagation*. A trained MLP can distinguish data that are not linearly separable.

3.2.2 Typical Architecture

Figure 3.5 illustrates an MLP with a single hidden layer (i.e., a shallow MLP). A shallow MLP is a function $f : \mathbb{R}^d \rightarrow \mathbb{R}^l$, where d is the size of the input vector $\mathbf{x}$ and l is the size

of the output vector $\mathbf{f(x)}$, such that, in matrix notation:

$$\mathbf{f(x)} = G(\mathbf{b}^{(2)} + W^{(2)}(H(\mathbf{b}^{(1)} + W^{(1)}\mathbf{x}))), \tag{3.19}$$

with bias vectors $\mathbf{b}^{(1)}$, $\mathbf{b}^{(2)}$; weight matrices $W^{(1)}$, $W^{(2)}$ and nonlinear activation functions H and G.

The vector $\mathbf{h(x)} = H(\mathbf{b}^{(1)} + W^{(1)}\mathbf{x})$ constitutes the hidden layer. $W^{(1)} \in \mathbb{R}^{d_h \times d}$ is the weight matrix connecting the input vector to the hidden layer of size d_h. Each row $W^{(1)}_{i\cdot}$ represents the weights from the input nodes to the i^{th} hidden node. Typical choices for H include $tanh$[2], and the logistic $sigmoid$ function[3] [78]. Both the $tanh$ and $sigmoid$ are scalar-to-scalar functions but their natural extension to vectors and tensors consists in applying them element-wise (e.g., separately on each element of the vector, yielding a same-size vector). Finally, the output vector is obtained as: $\mathbf{o(x)} = G(\mathbf{b}^{(2)} + W^{(2)}\mathbf{h(x)})$. The choice for G depends on the task in hand. For example, in binary classification $sigmoid$ is used whereas the typical choices for multiclass classification and regression are $softmax$ and identity functions, respectively.

3.2.3 Training MLPs with Backpropagation

To train an MLP, the set of parameters (i.e., $\{W^{(1)}, \mathbf{b}^{(1)}, W^{(2)}, \mathbf{b}^{(2)}\}$) should be learned. Given a set of training data, this can be achieved through the backpropagation algorithm.

The general idea is to boost the performance of a neural network by backward propagation of error signals, which relate the network's performance to its parameters. More specifically, the backpropagation algorithm finds the gradients of an error function E with respect to the parameters (i.e., weights and biases) of the network. These gradients are then used to update their corresponding network parameters. Therefore, each step of the training consists of three phases: (1) forward propagation of input through the network to

[2] $tanh(a) = (e^a - e^{-a})/(e^a + e^{-a})$
[3] $sigmoid(a) = 1/(1 + e^{-a})$

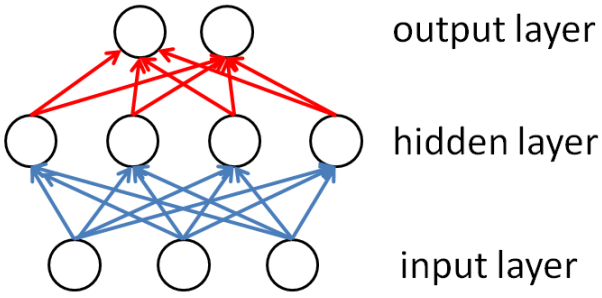


Figure 3.5: Illustration of a single-hidden-layer MLP (i.e., a shallow MLP) which is a function $f : \mathbb{R}^3 \rightarrow \mathbb{R}^2$. Weight matrices $W^{(1)}$ and $W^{(2)}$ are shown in blue and red respectively while bias vectors $\mathbf{b}^{(1)}$, $\mathbf{b}^{(2)}$ are not shown.

calculate activity of all nodes, (2) backpropagation of the error signals for calculating gradients, and (3) updating the network parameters using the gradients. These three phases should be repeated until the performance of the network is satisfactory or stops improving[4]. Next, I will give a brief description for each of these phases.

Forward propagation: The first phase of each training step is the forward propagation of a training pattern's input through the neural network in order to generate the corresponding activity of nodes (neurons) in all layers. For example, for neuron j in layer l of Figure 3.6 $z_j = H(a_j)$, where $a_j = \sum_i w_{ji} z_i$ is the activation received by neuron j.

Backward propagation: The second phase involves:

1. Backward propagation of the error signals through the neural network, using the training pattern target, in order to generate the partial derivative of the error function

[4]It is important to note this procedure is not guaranteed to find the global minimum of the error function [22].

with respect to the activation of all output and hidden neurons. For the output neuron k, $\frac{\partial E}{\partial a_k} = y_k - t_k$ where y_k is the response of the output neuron k and t_k is the corresponding target, for a particular input[5]. For the hidden neuron j, $\delta_j = H'(a_j) \sum_k w_{jk}\delta_k$ where $\frac{\partial E}{\partial a_j} \equiv \delta_j$ and $\frac{\partial E}{\partial a_k} \equiv \delta_k$ (see Figure 3.6).

2. Multiplying each weight's output delta with its input to get the gradient of the error function with respect to the weight[6]: $\frac{\partial E}{\partial w_{ji}} = \delta_j z_i$.

Weight update: The final phase of the training is weight update. For the quintessential gradient descent algorithm, this involves subtracting a ratio of the gradient (by multiplying it with the learning rate η) from the weight: $w_{ji}^{(\tau+1)} = w_{ji}^{(\tau)} - \eta \frac{\partial E}{\partial w_{ji}}$.

The reason for subtracting (not adding) a ratio of the gradient is that the gradient always indicates the increasing direction of the error function. So to reduce the error, the weight must be updated in the opposite direction (note that $\eta > 0$).

The learning rate η influences the speed and quality of learning. A larger learning rate leads to faster training whereas a smaller learning rate often results in more accurate training. Often, the learning rate is held fixed during the entire training. However, using a dynamic learning rate (instead of a fixed one) can increase the training efficiency[7].

3.2.4 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs or ConvNets) are variations of multilayer perceptrons that have been inspired more closely by biological processes [115]. CNNs have demonstrated substantial empirical success. Especially in object recognition tasks where CNNs have outperformed the hand-designed feature extraction approaches (e.g., SIFT and HOG)

[5]This rather concise partial derivative is the result of properly choosing the output activation G and error function E pair for each task (e.g., $softmax$ activation and cross-entropy error for multiclass classification).

[6]The same is true for the bias except that $z = 1$.

[7]That is because error surfaces usually consist of many flat regions as well as many extremely steep regions (see [227] for further explanation).

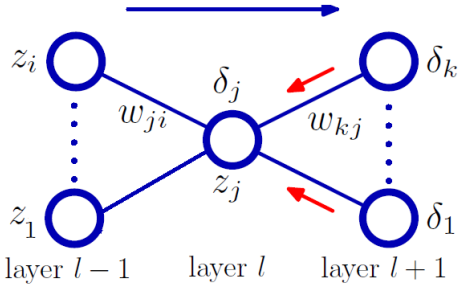


Figure 3.6: Illustration of the forward and backward propagations for hidden neuron j, from [22] with modification. For this neuron, δ_j is calculated by backpropagation of the δs from those units k to which unit j sends connections. The blue arrow denotes the direction of information flow during forward propagation, and the red arrows indicate the backward propagation of error information.

[71] and set the new state of the art for classification (e.g., MNIST dataset: 0.23% error [45]; ImageNet dataset: 2.99% error [146] vs. human error of 5.1% [179]). In addition to object recognition tasks, CNNs have been used in optic flow estimation [72], solving visual odometry [110], crowd segmentation [103], stereo matching [228], and action recognition from video sequences [101].

Typical Architecture

A typical CNN comprises several stages where the input and output of each stage are a set of (subsampled) feature maps (see Figure 3.7). Each stage is often composed of three layers: convolutional layer, nonlinearity layer, and pooling (subsampling) layer [115].

Convolutional Layer

Convolution, in the image processing context, means applying a kernel (a.k.a. filter) over an image at all possible offsets. For a two-dimensional image I and a two-dimensional

kernel K, the convolution can be mathematically described by:

$$C[i,j] = (I * K)[i,j] = \sum_m \sum_n I[m,n]K[i-m, j-n]. \tag{3.20}$$

Convolution can be used as a highly efficient method to describe transformations that apply the same linear transformations of a small, local region across the entire image [78]. Therefore, using convolution results in detecting features regardless of their position in the image. In CNNs, the connection weights between a single neuron of a feature map in stage s and a small patch (i.e., adjacent neurons) of a subsampled map in stage $s-1$ (or input image if $s = 1$) construct a kernel (see Figure 3.8). Also, since the same set of weights (i.e., kernel) connect all the neurons of a feature map to small patches of a subsampled map in $s-1$ (i.e., all possible offsets), the resulting connectivity functions as a convolution between the input feature map (or input image) and the kernel.

Nonlinearity Layer

All neurons in the convolutional layer have a point-wise nonlinear (e.g., sigmoid or rectified linear) function which is applied on the activation that they receive. In some convolutional networks, this point-wise nonlinearity is followed by a subtractive and divisive local normalization, which enforces local competition between adjacent features in a feature map, and between features at the same spatial location [115].

Pooling Layer

A pooling or subsampling layer adds robustness to the feature maps (in stage s) against small variations in the location of the features in the previous stage (i.e., $s-1$). More specifically, the pooling layer reduces the dimensions of a feature map by substituting non-overlapping patches in the map by their average (or maximum). Traditional CNNs apply a point-wise *tanh* after the pooling layer, but more recent models do not [115].

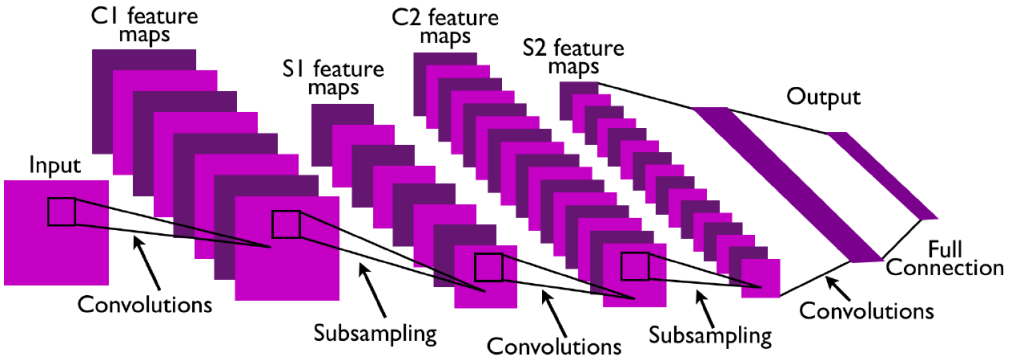


Figure 3.7: A typical CNN architecture with two feature stages followed by a fully-connected layer, from [115]. Each feature stage is composed of three layers: convolutional layer, nonlinearity layer (not shown in the figure), and subsampling layer.

Further Remarks

The sparse connectivity and weight sharing drastically reduce CNNs' parameters (i.e., number of unique connection weights) compared to a fully connected neural network. This reduction of free parameters increases learning efficiency (i.e., many fewer weights should be learned) and enables CNNs to achieve better generalization on vision problems by prevention of *overfitting*.

Although CNNs are conceptually simple, they come in quite versatile architectures with different number of stages and feature maps inside each stage. For a given pattern recognition task, a CNN learns to extract useful features at each stage. These features get more complex as information propagates through the network's stages [229].

Training CNNs with Backpropagation

The same procedure explained in 3.2.3 can be followed, with minor modifications, to train a CNN. These minor modifications should be made in order to ensure that the weight

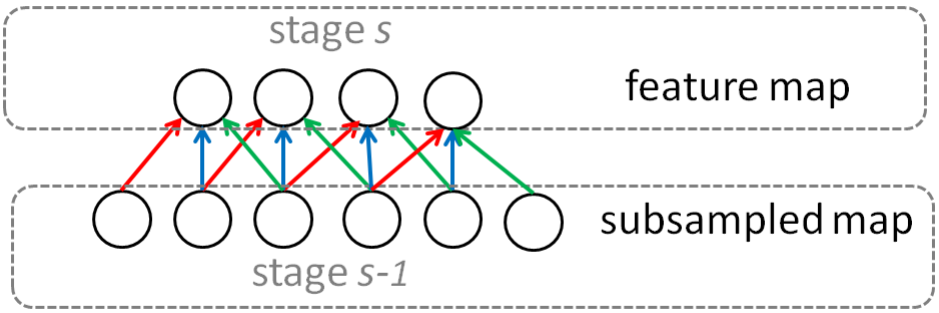


Figure 3.8: Sparse connectivity and weight sharing in CNNs (compare with the fully-connected network shown in Figure 3.5). Neurons of a subsampled map in stage $s-1$ are connected to neurons of a feature map in stage s. Weights of the same colour are shared (i.e., constrained to be identical). The weight vector (with red, blue, and green elements) construct a kernel (see Section 3.2.4).

sharing constraint in the convolutional and pooling layers are satisfied when evaluating the derivatives of an error function with respect to the adjustable parameters in the network.

Convolutional Layers

Since neurons within a feature map in stage s (indexed c_s) have different inputs but all share a common weight vector $\mathbf{w}^{(c_s)}$, errors $\delta^{(c_s)}$ from all neurons within the feature map will contribute to the derivatives of the corresponding weight vector. Therefore, the gradient of the error function with respect to each element of this vector is:

$$\frac{\partial E}{\partial w_i^{(c_s)}} = \sum_j \delta_j^{(c_s)} z_{ji}^{(p_{s-1})}, \tag{3.21}$$

where $w_i^{(c_s)}$ denotes the i^{th} element of the weight vector, $z_{ji}^{(p_{s-1})}$ represents the i^{th} input of the preceding pooling layer (indexed p_{s-1}) for the j^{th} neuron of feature map c_s and finally

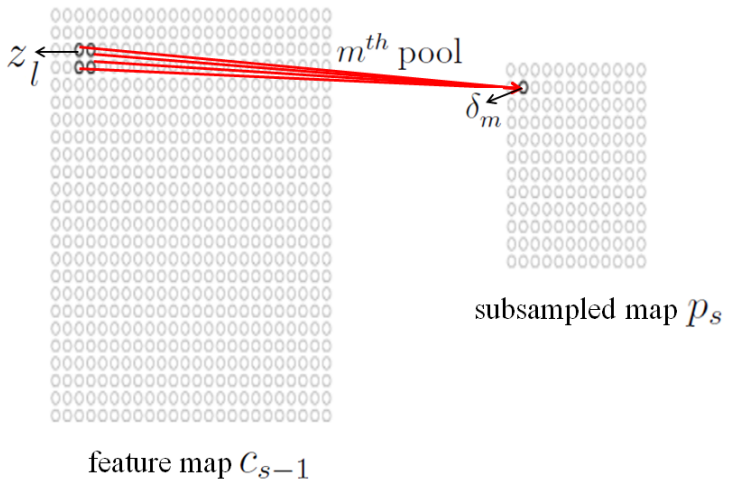


Figure 3.9: Illustration of a 2×2 pool in a pooling layer (see 3.2.4).

$\delta_j^{(c_s)}$ denotes the δ for j^{th} neuron of the same feature map which is computed recursively from δs of the neurons of the following layer [202].

Pooling Layers

As discussed earlier, pooling (or subsampling) layers can be either max-pooling or average pooling. In max-pooling layers, the error signal (δ_m in Figure 3.9) is only propagated through the neuron that had the maximum value of the pool in the forward propagation. In average pooling layers, the error signal is multiplied by the inverse of pool width times pool height (e.g., $\frac{1}{2\times2}$), and it is assigned to the whole pooling block so that all units get the same value.

Inspiration by and Analogy with Primate Visual Cortex

Inspiration

CNNs have their roots in the *neocognitron*, suggested by Fukushima in 1980 [116], which itself has been inspired by the model of the primary visual cortex (i.e., V1) proposed by Hubel and Wiesel.

More specifically, the idea behind convolutional layers are the simple cells in area V1 [78]. Due to the local wiring between retinal ganglion cells and simple cells, each simple cell only sees a portion of the visual scene (i.e., its receptive field) and can detect edges (i.e., features) within that portion. Like simple cells, each artificial neuron in a feature map only receives input from some of the neurons in its preceding map and detect features in a limited area of the preceding map. But due to weight sharing, the same features across the entire image are extracted, though by different neurons in the feature map.

Additionally, pooling layers, which help the networks to tolerate local shifts of features, have been inspired by complex cells of V1 [78]. Complex cells receive inputs from many simple cells and have local invariance to the location of a feature inside their receptive fields.

Finally, local contrast normalization between adjacent features in a feature map [170], and between features at the same spatial location, in the nonlinearity layer, has been inspired by the *lateral inhibition* [23] and *cross-orientation inhibition* [140] models of the primary visual cortex [18]. These models explain how each excited cell reduces the activity of its neighbors, resulting in the sharply tuned orientation-selectivity of V1 neurons.

Analogy

As explained in Section 3.2.4 all neurons in a feature map are followed by point-wise nonlinearities. This is analogous to how computational neuroscientists model the firing

rate of a neuron as the output of a static nonlinearity applied on the current generated in the neuron's soma [65].

Receptive fields of V1 simple cells can be mathematically modelled as Gabor filters that act as local filters selectively responding to edges in the visual scene [97]. Interestingly, if a CNN is trained with natural images, its earliest convolutional layers resemble the receptive fields of V1 simple cells.

Furthermore, [225] have recently shown that a performance-optimized CNN can explain the neural encoding in higher ventral areas; a fundamental open question in systems neuroscience. Specifically, they trained a CNN for a challenging high-variation object recognition task. Even though the network was never explicitly constrained to match neural data, its output layer was highly predictive of neural responses in the inferior temporal (IT) cortex, better than any other IT model. Moreover, the middle layers of the model were highly predictive of V4 neural responses, suggesting top-down performance constraints directly shape intermediate visual representations [225].

3.2.5 LSTM Networks

A shortcoming of convolutional neural networks (or any feedforward neural network) is that they lack memory. Memory is essential for correct inference in problems where input is a sequence of observations rather than just one e.g., action recognition in a video (a sequence of frames) vs. object recognition in an image.

Recurrent neural networks (RNNs) have been introduced for such sequence prediction problems. The basic idea behind RNNs is simple: by adding feedback to a feedforward network, we can build a network that at each time step receives both the current value in the sequence and its own previous output (calculated on the preceding value in the sequence). This feedback connection allows the network to maintain information about previous input values in the sequence. In practice, such RNNs fall short of learning long-term dependencies due to the well-known problem of vanishing/exploding gradients [19].

To address this shortcoming, Hochreiter and Schmidhuber [91] proposed long short-term memory (LSTM) networks, which can remember relevant information for long periods of time while forgetting irrelevant information. More specifically, each LSTM unit is equipped with a system of gating units that controls the flow of information. This gating system allows the time scale of information integration to dynamically change based on context because the integration time constants are output by the cell itself not fixed parameters [78].

Since the introduction of LSTMs, many papers have been published improving them and quite a few different variations have been suggested [74, 70, 75]. A widely-used variation of LSTMs, proposed by Gers et al. [74], is composed of an input gate, an output gate and a forget gate. Figure 3.10 depicts an LSTM unit (cell). A network that contains at least one such unit is called an LSTM network. At time step t, in addition to the main input $\mathbf{x}_t$ (activity vector of the layer that precedes the LSTM in the network), the LSTM unit receives two other input vectors $\mathbf{c}_{t-1}$ (cell state at the previous time step) and $\mathbf{h}_{t-1}$ (output at the previous step). Given these inputs, the unit generates the cell state at the current step $\mathbf{c}_t$ and the output $\mathbf{h}_t$.

More formally, the forget gate's activation vector $\mathbf{f}_t \in \mathbb{R}^h$ (where h is the number of hidden nodes inside the LSTM unit) at time t is computed as,

$$\mathbf{f}_t = \sigma\big(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + \mathbf{b}_f\big), \tag{3.22}$$

where σ denotes sigmoid function, $W_f \in \mathbb{R}^{h \times d}$ (where d is the number of features to the LSTM unit), and $U_f \in \mathbb{R}^{h \times h}$ are the weight matrices and $\mathbf{b}_f \in \mathbb{R}^h$ is the bias. The input gate's activation vector $i_t \in \mathbb{R}^h$ at time t can be calculated as,

$$\mathbf{i}_t = \sigma\big(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + \mathbf{b}_i\big), \tag{3.23}$$

where $W_i \in \mathbb{R}^{h \times d}$, and $U_i \in \mathbb{R}^{h \times h}$ are the weight matrices and $\mathbf{b}_i \in \mathbb{R}^h$ is the bias. The output gate's activation vector $\mathbf{o}_t \in \mathbb{R}^h$ at time t is computed as,

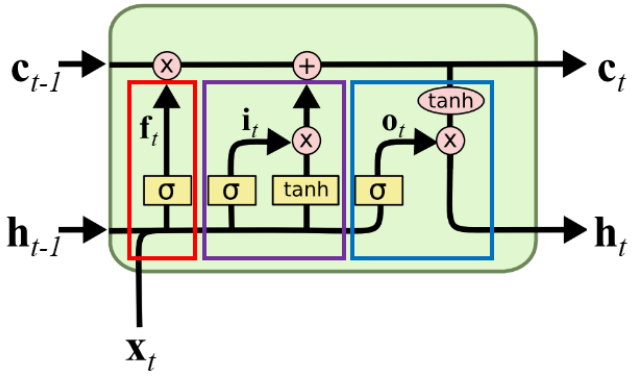


Figure 3.10: Long short-term memory (LSTM) unit. A typical LSTM composed of an input gate (purple box), an output gate (blue box) and a forget gate (red box). The input gate decides which new values flow into the unit, the forget gate decides which values remain in the unit, and finally the output gate decides which parts of the cell state $\mathbf{c}_t$ are used to compute $\mathbf{h}_t$ the output vector. $\mathbf{x}_t$, $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ are, respectively, input vector from the preceding layer to the LSTM unit, forget gate's activation vector, input gate's activation vector, and output gate's activation vector.

$$\mathbf{o}_t = \sigma\big(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + \mathbf{b}_o\big), \tag{3.24}$$

where $W_o \in \mathbb{R}^{h\times d}$, and $U_o \in \mathbb{R}^{h\times h}$ are the weight matrices and $\mathbf{b}_o \in \mathbb{R}^h$ is the bias. The cell state's vector $\mathbf{c}_t \in \mathbb{R}^h$ at time t is computed as,

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ tanh\big(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1} + \mathbf{b}_c\big), \tag{3.25}$$

where $\circ$ denotes entry-wise product, $W_o \in \mathbb{R}^{h\times d}$, and $U_o \in \mathbb{R}^{h\times h}$ are the weight matrices and $\mathbf{b}_o \in \mathbb{R}^h$ is the bias. Finally, the output vector $\mathbf{h}_t \in \mathbb{R}^h$ at time t is computed as,

$$\mathbf{h}_t = \mathbf{o}_t \circ tanh(\mathbf{c}_t), \tag{3.26}$$

where $tanh$ denotes the hyperbolic tangent function.

LSTM networks are trained using backpropagation through time (BPTT). BPTT begins by unfolding the network in time. The unfolded network contains τ copies that share the same parameters, where τ is the number of observations in the sequence. Next, the backpropagation algorithm is used to find the gradient of the cost with respect to all the network parameters (see Section 3.2.3).

3.2.6 Dropout

Dropout, proposed by Srivastava et al. [199], is a computationally-cheap regularization technique that prevents overfitting in deep neural networks. At every step of training, each hidden unit is randomly omitted (dropped) from the network with a determined probability. This approach prevents the hidden units from learning correlated features that are only useful together and more likely represent noise [78]. During inference (test time) all hidden neurons are used (no dropouts), while their activities are scaled so that the overall magnitude of the hidden neurons stays the same as training time.

3.2.7 Batch Normalization

Batch Normalization (BN), proposed by Ioffe and Szegedy [99], is an effective technique not only for improving the performance of neural networks but for speeding up their training. The idea is to normalize (transform) the activities of each hidden layer such that they have a fixed mean and variance, which are learned during training. More formally, we can write this transformation of a hidden unit's activities within a mini-batch x_is $\rightarrow$ y_is as,

$$y_i = \gamma \frac{x_i - E[x_i]}{\sqrt{Var[x_i] + \epsilon}} + \beta \tag{3.27}$$

where $E[x_i]$ and $Var[x_i]$ represent the mini-batch mean and variance, ϵ is a small number (e.g., 1e-5) to avoid division by zero, and γ and β are trainable parameters.

BN effectively reduces an undesirable phenomenon, called internal covariate shift. This covariate shift exists because as the parameters of a layer change during training, the input distribution of all its following layers will change too [99].

BN also decreases overfitting because it has regularization effects. Namely, activation of each hidden unit is first subtracted and then multiplied by random values (mini-batch's mean and standard deviation) at each step of training. Therefore, each layer learns to be robust to variation in its input.

Batch Normalization at Test Time

At test time, we might only have one sample (not a mini-batch) to work with. This means that we no longer have a mini-batch to calculate the mean and variance from. Instead, we estimate the mean and variance of the whole dataset (population) during training. The most popular approach to estimate these values is to use an exponential moving average.

Combining Dropout and Batch Normalization

While some papers (e.g., [85]) suggest that using Batch Normalization (BN) together with Dropout achieves a better performance, Li et al. [118] showed both theoretically and numerically that combining the two often leads to a worse performance. Indeed, my experiments (on the networks that I explain later in the thesis) agree with their finding: the best performance has been achieved using only BN.

Concluding Remark

So far, I discussed the related neuroscience background (Chapter 2) and computer-vision tools (Chapter 3), which have been used in the thesis. Next, I will describe the empirical model that I propose for area MT.

Chapter 4

A Video-Driven Model of Response Statistics in the Primate Middle Temporal Area

4.1 Introduction

The middle temporal cortex (MT) receives strong feedforward input from early visual areas V1, V2, and V3 [130, 126], as well as direct sub-cortical input [194, 25]. It projects to the higher-level middle superior temporal and ventral intraparietal areas, and also receives strong feedback connections from these. Electrical stimulation of MT affects perception of visual motion [147]. Inactivation or damage of MT impairs motion perception [143, 178] and the ability to smoothly follow a moving object with the eyes [144]. Illusions in speed perception have also been linked with subtle properties of MT neuron responses [27].

Consistent with these effects, many neurons in MT respond strongly to visual motion. The spike rates of individual MT neurons vary with a number of stimulus features, including direction and speed of visual motion, and binocular disparity. Many MT neurons are

sensitive to motion in depth, i.e., toward or away from the eyes [51]. MT is the earliest visual region in which a substantial number of neurons solve the motion "aperture problem", responding to the actual direction of motion of a stimulus, rather than the component of motion that is orthogonal to local edges, which requires only local computations [154, 197]. In summary, MT exhibits a particular representation of visual motion, which is similar in scope to scene flow [134].

Although much is known about this representation, and its causal role in visual motion perception, some aspects of the relationship between the representation and ethologically relevant functions are less clear. For example, the accuracy of smooth-pursuit eye movement, self-motion perception, and motion-based segmentation may be sensitive to particular tuning properties or population statistics, in addition to artificial disruptions of MT activity. Computational models can be used to study such relationships, and sophisticated computational models of MT responses have been developed [148, 13]. However, I wondered if a new model could be developed that spans a more comprehensive range of MT response phenomena, and captures MT response statistics in more detail. Rather than building on existing mechanistic models of MT, I instead pursued an empirical model, in which I directly specify the neurons' tuning curves. This approach allows me to approximate the response statistics in almost arbitrary detail, without requiring a complete understanding of how these responses arise in the brain.

4.2 Methods

4.2.1 Structure of the Empirical Model

The proposed model produces approximations of MT spike rates directly from input video. I focus on producing spike rates, rather than spike sequences. As an aside, given these rates, it is straightforward to produce Poisson spike sequences [52], including those with noise correlations that are realistic for MT [212].

The model structure is sketched in Figure 4.1. The model requires five fields as input. The field values are defined at each image pixel x, y. The five fields are $u(x, y)$ (horizontal flow velocity), $v(x, y)$ (vertical flow velocity), $d(x, y)$ (disparity), $c(x, y)$ (contrast), and $a(x, y)$ (attention). Section 4.2.2 below discusses calculation of these fields.

The response of each neuron is approximated as a nonlinear-linear-nonlinear (NLN) function of these fields. The first nonlinear step requires calculation of four additional fields for each neuron, each of which is a point-wise nonlinear function of the five input fields. I refer to these functions as tuning functions (see details in Section 4.2.3). Each of these tuning functions is used to scale the neuron's response to a different stimulus feature. Specifically, I calculate $g_s(u, v, c)$ (a function of flow speed and contrast), $g_\theta(u, v)$ (a function of flow direction), $g_d(d)$ (a function of disparity), and $g_g(a, c)$ (a function of attention and contrast). Whereas the first five fields are correlates of MT responses (e.g., velocity), these additional fields represent nonlinear tuning functions of these correlates. In the excitatory part of a unit's receptive field, each of these fields has a monotonic relationship with spike rates when other fields are held constant.

The full model therefore requires calculation of four times as many of these tuning-function fields as there are neurons with distinct sets of parameters. The model has uniform response statistics across the visual field (similar to convolutional networks), so there is one such set of parameters per distinct response channel in the MT layer. This number can be specified at run time, but I would expect it to normally be on the order of 100-1000, therefore 400-4000 of these fields must be calculated by the full model. One additional field per neuron is then calculated as the point-wise product of these fields (consistent with data from [174, 211]). I refer to this as the neuron's tuning field,

$$t(x, y) = g_s g_\theta g_d g_g. \tag{4.1}$$

This completes the first nonlinear stage of the NLN model. Similar to convolutional networks, only one tuning field is needed per channel (feature map), corresponding to a set of model parameters, regardless of the pixel dimensions of the channel. Henceforward, when I talk about a "neuron model", it should be understood that this "neuron model" is

ultimately tiled across the visual field to simulate many neurons with different receptive field centres.

The remaining linear and nonlinear steps consist of a conventional convolutional layer, with one channel per MT neuron (I specify the number of MT neurons at instantiation time, and choose parameters for each one as discussed below in Section 4.2.4). Kernels combine tuning-field values $t(x, y)$ over a receptive field. However (in contrast with typical convolutional layers with learned kernels), kernels are parameterized to resemble MT receptive fields. The kernels include excitatory, direction-selective suppressive, and non-selective suppressive components. Such components have been found to account well for MT responses to complex motion stimuli [49]. The excitatory component of the kernel models the neuron's classical receptive field. This component has positive weights and a Gaussian structure, which is elongated so that the axis of elongation is orthogonal to the neuron's preferred direction [167]. It spans a single channel of the tuning-field layer, and therefore has a speed and direction selectivity that match that channel. The direction-selective suppressive component also spans a single tuning-function channel. It has negative weights, and is also modelled as a Gaussian function. Relative to the excitatory kernel, it can be symmetrically larger, or elongated, or offset. For each neuron, I draw at random from these spatial relationships with the proportions reported by Xiao et al. [223]. The preferred direction of this suppressive component is generally different from that of the excitatory component. I draw this difference from the distribution in Cui et al. [49] (their Figure 5). Finally, the non-direction-selective suppressive component receives the same tuning-function channel with g_θ removed. It has negative weights and an annular structure that I model as a rectified difference of Gaussians. The full kernel is the sum of these components. When I fit tuning curves for speed, disparity, and direction tuning in response to stimuli that are spatially uniform in these properties, I simplify the kernels as broad Gaussian functions.

The final nonlinearity is,

$$f(x) = [Ax + B]_+^n \,, \tag{4.2}$$

composed of a half-wave rectification ($[]_+$) followed by a power function ($[]^n$). A and B are a scaling factor and a background spike rate, respectively.

I have chosen this form for the proposed model (versus other possible forms with different orders of the linear and nonlinear parts), because the linear kernel must follow at least some of the tuning curves for consistency with data from Majaj et al. [125] (see Figure 4.5). Also to avoid negative spike rates due to inhibitory surrounds, the final rectifying nonlinearity must come after the linear kernel.

Eccentricity and Receptive Field Size

The visual cortex differs from convolutional networks in that the receptive fields of neurons in many visual areas scale almost linearly with eccentricity (visual angle from the fovea). This difference could be reduced by remapping the input images. However, to simplify use of the model with standard uniform-resolution videos, I instead model the whole visual field uniformly, as is typical in convolutional networks. There is also variation in receptive field sizes at any given eccentricity. I modelled the spread of receptive field sizes on parafoveal receptive fields (2-10 degree eccentricity) from Figure 2 of Maunsell and Van Essen [132].

4.2.2 Input Fields

The model requires contrast, attention, optic flow, and binocular disparity fields.

Contrast

The contrast field is calculated using the definition of Peli [160]. This is a local, band-limited measure, in contrast with other notions of contrast (e.g., root-mean-squared luminance) that are global and frequency-independent. A local definition is needed to modulate neuron responses according to contrast within their receptive fields (as opposed to remote

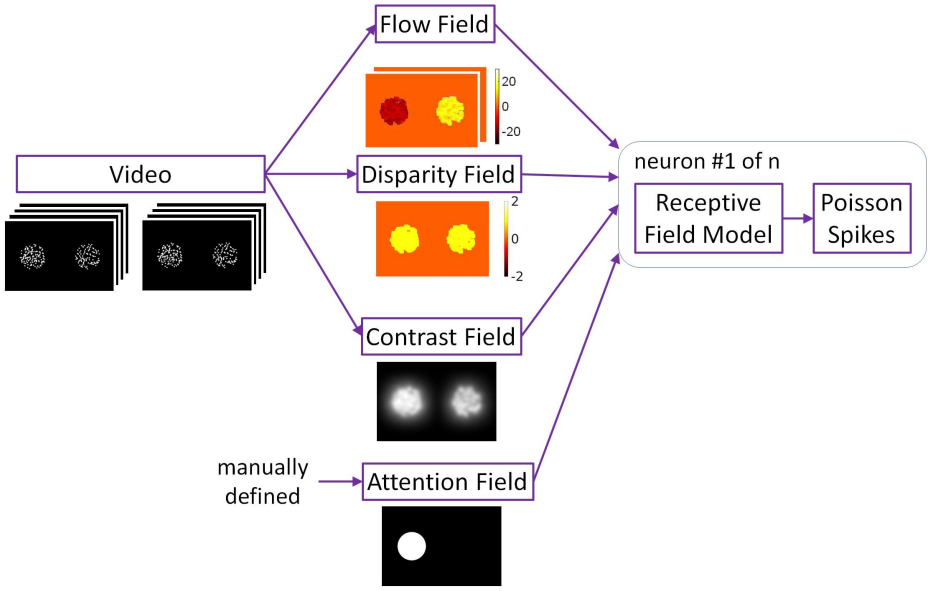


Figure 4.1: Structure of MT model. The model uses nonlinear-linear-nonlinear models to approximate neuron responses as functions of optic flow, contrast, disparity, and attention fields. Optic flow, contrast, and disparity are calculated from input images, as described in the text. An example of these fields can be seen for a video input with two patches of random dots moving in opposite directions (i.e., up and down; with far disparity) where the left patch was attended. Units for flow and disparity maps are deg/sec and deg. Poisson spikes can optionally be generated at the estimated spike rates to emulate neural activity more closely, but they are not used in this thesis.

parts of the image). Frequency dependence allows me to match the contrast definition to primate contrast sensitivity [173, 53].

In Peli's definition, contrast at each spatial frequency band (i) is defined as a ratio of two functions,

$$c_i(x,y) = \frac{\alpha_i(x,y)}{l_i(x,y)}. \tag{4.3}$$

The numerator function is,

$$\alpha_i(x,y) = I(x,y) * g_i(x,y), \tag{4.4}$$

where I is the image, g_i is a spatial frequency dependent filter, and $*$ denotes convolution. The denominator function is,

$$l_i(x,y) = \bar{I} + \sum_{j=1}^{i-1} \alpha_j(x,y), \tag{4.5}$$

where $\bar{I}$ is the image mean. Peli suggested cosine log filters as the choice for g_is since an image filtered by a bank of these filters can be reconstructed by a simple addition process without distortion. However, to relate the contrast definition more directly to V1, I instead used a bank of Gabor filters with four different frequencies and four different orientations for a total of 16 contrast channels. I combined these channels in a weighted sum:

$$c'(x,y) = \sum_{k=1}^{16} A_k c_k(x,y), \tag{4.6}$$

where $A_k s$ were chosen to approximate macaque contrast sensitivity [53, 54].

I then smoothed the resulting contrast field with a 2D Gaussian kernel, which was meant to approximate integration over V1 cells, and scaled it so that its mean over the image was equal to the root-mean-squared contrast measure:

$$c(x,y) = A_{scale} gaussfilt(c'(x,y)). \tag{4.7}$$

Attention

Attention is typically driven by task demands, so in general it can not be derived from images alone (in contrast with saliency). Recent models approximate top-down influences [24]. However, in the context of training neural networks that have attention mechanisms (e.g., [224]), the attention field should ideally be defined by the network itself, to align attention modulation of activity with the network's focus of attention. Therefore I treated the attention field as an input to the model. To test the model, and to compare its output with electrophysiology data, I manually defined attended stimulus regions by drawing polygons around them in a custom user interface.

Flow and disparity fields

Flow and disparity fields were calculated using computer-vision algorithms. Specifically, I used the Lucas-Kanade method [123] to estimate both optic flow and disparity from images. This generally produced good fits to MT data (see Results).

The classical Lucas-Kanade algorithm does not capture large displacements, but this limitation is addressed by a multi-scale version of the algorithm [129]. In this version, the Gaussian pyramids method is used to repeatedly halve the image resolution. Flow or disparity is then estimated at the lowest resolution first. Then at each finer resolution, the immediate lower-resolution estimate is used to warp the earlier image, and the Lucas-Kanade algorithm is used to find residual differences between the warped earlier image and the later image. The multi-scale version of the algorithm also helps to solve the aperture problem, since it finds estimates that are consistent with global motion apparent in downsampled images. I typically used the multiscale algorithm in the simulations, with 3-5 scales. To simulate combined local and pattern motion selectivity [154], I mixed the outputs of single-scale and multi-scale versions of the algorithm.

I also explored a variety of other algorithms for flow and disparity estimation, including semi-global matching [90], Classic++ [201], loopy belief propagation on a Markov random

field [69], and a convolutional neural network Žbontar and LeCun [228]. Several of these methods extrapolated far beyond well-textured regions, e.g., reporting motion over the whole image in response to a small stimulus. I interpreted this as being physiologically unrealistic, because it involves lateral communication over the whole visual field. However it does not actually expand the units' classical receptive fields unrealistically, because there is no response at zero contrast (see Equation 4.14). For the experiments, I used the Lucas-Kanade with pyramids, because it is simple and well established, and I did not find other methods to provide substantial advantages within the scope of this thesis. However, future work may reveal such advantages.

4.2.3 Tuning Functions

Given these fields, the next step in approximating a neuron's activity was calculation of a new four-channel image that consisted of pixel-wise nonlinear functions of the fields. Specifically, I calculated $g_s(u, v, c)$ (a function of flow speed and contrast), $g_\theta(u, v)$ (a function of flow direction), $g_d(d)$ (a function of disparity), and $g_g(a, c)$ (a function of attention and contrast). These functions were adopted from previous studies, as described below.

Speed Tuning

I used a contrast-dependent speed tuning function, [149],

$$g_s = \exp\left(-\frac{[\log\left(q(s,c)\right)]^2}{2\sigma_s^2}\right), \tag{4.8}$$

where,

$$q(s,c) = \frac{s + s_0}{s_p(c) + s_0}, \tag{4.9}$$

$s = \sqrt{u^2 + v^2}$ is motion speed, s_p is the preferred speed. The tuning curve has parameters s_0 (offset) and σ_s (width). Preferred speed is a function of contrast,

$$s_p(c) = \frac{A_p c}{c + B_p}, \tag{4.10}$$

where c is contrast at each pixel (Equation 4.7) and A_p and B_p are additional parameters that define a saturating dependence of preferred speed on contrast.

When stimulated with sinusoidal gratings, about a quarter of MT neurons show selectivity for certain spatial and temporal frequencies, rather than speed (defined as the ratio between spatial and temporal frequencies) [164]. Another quarter of MT neurons are selective to grating speed, regardless of its spatiotemporal components, and the remaining neurons form a continuum between these two behaviours. A similar distribution is also observed in V1 [165]. However, more complex stimuli containing a broader spectrum of frequencies, e.g., random dot fields, elicit in MT selective responses to speed. Since my goal was to apply this model on naturalistic stimuli, which have broad frequency contents, I included speed tuning and ignored selectivity for spatial and temporal frequencies in the model.

Direction Tuning

Direction tuning was modelled as [220],

$$g_\theta = \exp\left(\frac{cos\left(\theta - \theta_p\right) - 1}{\sigma_\theta}\right) + a_n \exp\left(\frac{cos\left(\theta - \theta_p - \pi\right) - 1}{\sigma_\theta}\right), \tag{4.11}$$

where $\theta = atan2(v, u)$ is motion direction, θ_p, σ_θ, and a_n are the preferred direction, direction width, and relative amplitude in null direction (i.e., 180 degrees away from preferred direction), respectively.

Disparity Tuning

Similarly, disparity tuning was modelled using Gabor functions [56],

$$g_d = \exp\left(\frac{-\left(d - d_p\right)^2}{2\sigma_d^2}\right) \cos\left(2\pi f_d(d - d_p) + \phi_d\right), \tag{4.12}$$

where d_p and σ_d set the centre and width of the Gaussian component and f_d and ϕ_d are the frequency and phase of the oscillatory component.

Attention and Contrast

Lastly, the gain function was [211, 128],

$$g_g(a, c) = \begin{cases} A_g g_c(c), & \text{if } a = 1 \\ \\ g_c(c), & \text{if } a = 0 \end{cases} \tag{4.13}$$

where A_g is the attentional gain and g_c, is the contrast response function defined as:

$$g_c(c) = \frac{A_c c^n}{c^n + B_c}, \tag{4.14}$$

where A_c and B_c are the contrast gain, contrast offset, contrast exponent, and c is contrast at each pixel (Equation 4.7).

Binocular Interactions

In many of the electrophysiology experiments that inform the model, monkeys were free to converge their eyes on a single, flat computer display, with constant (near zero) binocular disparity. However in a more complex environment, some MT neurons are tuned for motion-in-depth [51]. To account for such 3D motion encoding of MT neurons, I extended the proposed model by modifying Equation 4.2 as,

$$f(x) = [A_L x_L + A_R x_R + B]_+^n , \tag{4.15}$$

where A_L and A_R are left and right eye gains, and x_L and x_R are weighted sums of tuning functions in left and right eye respectively.

A limitation is that the model of motion-in-depth is not realistically integrated with the model of binocular disparity. To retain realistic disparity tuning, I simply used the disparity tuning field, and identical disparity tuning curves in each eye, so that disparity and motion-in-depth tuning are orthogonal.

4.2.4 Model Fitting

Tuning Curve Fits

To test the model, I fit various tuning curves from the electrophysiology literature using Matlab's nonlinear least-squares curve fitting function, *lsqcurvefit* (trust-region-reflective algorithm). The fitting procedure for a given tuning curve selected the parameters of the relevant tuning functions (e.g., $g_s(u, v, c)$), along with parameters A and B of Equation 4.2. As the optimization was non-convex, I initiated it from at least 100 different starting points for each neuron, and took the most optimal answer.

This approach was designed to have a high success rate, in order to reliably support development of a rich statistical model of MT activity. Aside from failures of the optimization procedure (which I minimized by restarting from many initial parameter values), the approach has two potential failure modes. The first would arise from a poor choice of nonlinear function, however I chose functions that are well supported by previous work. The second would be a failure of the computer vision algorithms to estimate the relevant parameters from the images. I generally had good results with the Lucas-Kanade algorithm (see Results).

Parameter Distributions

I drew the neurons' tuning parameters from statistical distributions that were based on histograms and scatterplots in various MT electrophysiology papers. The model required distributions of preferred disparity, preferred speed, speed-tuning width, attentional index [211], and a number of other tuning properties. As a first step in approximating these distributions, I extracted histograms and scatterplots of various tuning properties from the literature using Web Plot Digitizer (https://automeris.io/WebPlotDigitizer/). I then modelled each histogram using either a standard distribution (one of Gaussian, log-Gaussian, Gaussian mixture, gamma, t location-scale, exponential, and uniform), or the Parzen-window method [157]. For Parzen-window method, I selected the bandwidths using

Silverman's rule of thumb [192]. In each case, I chose the distribution model that minimized the Akaike Information Criterion [3]. The parameter distributions are summarized in Table 4.1.

Correlation between Model Parameters

To make the proposed model more realistic, I looked for studies that examined the correlation between the tuning parameters in area MT. Bradley and Andersen [29] found that the centre-surround effects of disparity and direction are mainly independent of each other, supporting the way I combine them over the MT receptive field. In another study, DeAngelis and Uka [56] did not find a correlation between direction and disparity tuning parameters. They reported a non-zero correlation between speed and disparity tuning (neurons with higher speed preference tend to have weak and broad disparity tuning). However, this correlation was weak (see their Figure 11.A) and therefore I ignored it in the proposed model. They also found a correlation between the preferred disparity and the disparity phase of the neurons whose preferred disparity is close to zero. I included this correlation by modelling the conditional distribution of disparity phase given the preferred disparity.

4.2.5 Dynamics of Component and Pattern Selectivity

The neurophysiology of the aperture problem in optic flow has been studied with overlapping pairs of drifting sine-wave (or square-wave) gratings at different angles, which together form a percept of a plaid pattern moving in an intermediate direction. MT is the earliest visual area to solve the aperture problem, in the sense that many MT neurons respond to the direction of the plaid pattern rather than the sinusoidal components [142, 216]. More specifically, studies in alert monkeys have shown that direction selectivity in many MT neurons evolves over time such that they become selective to the direction of the pattern as opposed to direction of the components [200, 154]. On the other hand,

Table 4.1: Distribution families used for various tuning parameters, and sources in the literature from which distributions were estimated. The number in the bracket specifies the dimension of a parameter, for those that have more than one.

Parameter	Distribution	Source
Preferred direction	Uniform	DeAngelis and Uka [56]
Direction bandwidth	Gamma	Wang and Movshon [220]
Null-direction amplitude	t location-scale	Maunsell and Van Essen [130]
Preferred speed	Log uniform	Nover et al. [149]
Speed width	Gamma	Nover et al. [149]
Speed offset	Gamma	Nover et al. [149]
Attentional index	t location-scale	Treue and Martínez Trujillo [211]
Contrast influence on preferred speed (2)	2D Gaussian mixture	Pack et al. [156]
Contrast influence on gain (3)	Conditional on attentional index	Martınez-Trujillo and Treue [128]
Preferred disparity	t location-scale	DeAngelis and Uka [56]
Disparity frequency	Log normal	DeAngelis and Uka [56]
Disparity phase	Gaussian mixture (two components)	DeAngelis and Uka [56]
Ocular dominance	t location-scale	DeAngelis and Uka [56]
CRF size	t location-scale	Maunsell and Van Essen [132]

studies in anesthetized monkeys reached conflicting results where one study [155] reported that pattern selectivity of MT neurons was significantly impaired in anesthetized animals (where only 7% of MT cells were pattern selective as opposed to 60% in alert animals) while several other studies [142, 175, 164, 197] reported the same proportions of pattern selective neurons in MT as observed in alert animals. Among these studies, Smith et al. [197] conducted the most comprehensive experiment to investigate the MT neural response dynamics by examining the responses of 143 MT neurons over cumulative time windows, and reporting the Z-transformed pattern- and component-response correlations (Z-scores). They classified each of the cells, based on their Z-scores in the last time window, as pattern direction selective, component direction selective, or "unclassified".

Based on this study, the proposed model approximates the distributions of pattern and component selectivity in each time window, and also realistic trajectories of the mean selectivities of each category of cells. To reproduce this behaviour, I first fit 2D Gaussian distributions to scatterplots of pattern and component selectivity (Figures 3 and 5 of Smith et al. [197]). To create a model of an n-neuron population, I drew n samples from the distribution for each time window. Then, to model each cell, I grouped together one pattern/component selectivity sample from each time window, as follows. Starting from the final time window, I classified the pairs to one of the three classes (pattern, component, or unclassified, as in [197]). Then I used the Hungarian algorithm [113] to match each sample in the second-last time window with a sample in the last time window. The match minimized the total of Euclidean distances between matched pairs of samples, except that I perturbed these distances with Gaussian noise, 0 +/- 2.5SD, to reproduce overlap between groups in the second-last time window. I continued this assignment process backwards in time until the pairs of the first time window were assigned to those of the second.

I produced responses with specified pattern and component correlations by combining pure pattern and component responses. To do this, I began by drawing a direction-tuning width sample. I then calculated the correlation between the pattern and component responses, r_{pc} (which depends on the direction-tuning width), and I calculated the partial pattern and component correlations R_p and R_c from the corresponding Z-scores. I

then constructed a new signal $S_t = F(S_c, S_p, \boldsymbol{p})$ where F is a function of the component-direction-selective response (S_c), pattern-direction-selective response (S_p), and a vector of parameters $\boldsymbol{p}$. I found the parameters $\boldsymbol{p}$ in an optimization process whose objective was to fit the partial pattern and component correlations (R_p and R_c).

I tried the simple additive form for F:

$$S_t = F(S_c, S_p, \boldsymbol{p}) = p_1 S_c + p_2 S_p, \tag{4.16}$$

but this gave poor results. I therefore considered three other forms,

1. Multiplicative, $S_t = F(S_c, S_p, \boldsymbol{p}) = p_1 S_c + p_2 S_p + p_3 S_c S_p$,
2. Expansive $S_t = F(S_c, S_p, \boldsymbol{p}) = p_1 S_c + p_2 S_p + p_3 (S_c + S_p)^2$,
3. Compressive $S_t = F(S_c, S_p, \boldsymbol{p}) = p_1 S_c + p_2 S_p + p_3 (S_c + S_p)^{.5}$.

(see Results for comparison).

4.2.6 Comparison With Previous Models

I compared tuning curves of the proposed model to the models of Nishimoto and Gallant [148] and Baker and Bair [13], with some modifications. I chose these models because they are recent and video-driven. Both build on a previous influential MT model [180]. Below I describe my adaptations of these models. Note that I only use these models to provide points of comparison with my empirical model, which is otherwise unrelated.

In the model of Nishimoto and Gallant [148], a video sequence first passes through a large bank of V1-like spatiotemporal filters with rectifying nonlinearities. The filter outputs are combined over local neighbourhoods through divisive normalization. Finally, the normalized outputs are weighted optimally to approximate neural data.

As in Nishimoto and Gallant [148], I used a bank of $N = 1296$ filters, including those with spatial frequencies up to two cycles per receptive field. In a departure from Nishimoto

and Gallant [148], I used multivariate linear regression to optimize the weights, as in Rust et al. [180]. More specifically, to optimize the weights, I generated training and testing movies for each tuning curve. Each movie was $2000 \times M$ frames in length, where M was the number of data points in the tuning curve. I used the training movie as input to the model and found the weights that minimized the error function,

$$E(\mathbf{w}) = \|\mathbf{X}_{train}\mathbf{w} - \mathbf{R}\|^2 + \lambda\|\mathbf{w}\|^2, \tag{4.17}$$

where $\mathbf{w} \in \mathbb{R}^{10\mathrm{N}}$ is the weight vector, $\mathbf{X}_{train} \in \mathbb{R}^{2000\mathrm{M}\times 10\mathrm{N}}$ is a matrix containing normalized V1 responses (from the spatiotemporal filters) when the training movie was used as input, $\mathbf{R} \in \mathbb{R}^{2000\mathrm{M}}$ is a vector containing the MT responses, and λ is a regularization constant. The optimal weights that minimize this error function can be computed from,

$$\mathbf{w} = \left(\mathbf{X}_{train}^T\mathbf{X}_{train} + \lambda I\right)^{-1}\mathbf{X}_{train}^T\mathbf{R}, \tag{4.18}$$

where T denotes matrix transpose, -1 denotes matrix inverse, and I denotes the identity matrix.

The model of Baker and Bair [13] is composed of two cascaded circuits. The first circuit calculates the motion response while the second calculates disparity. However, they used only the first circuit to approximate the motion tuning of MT neurons. I implemented their motion circuity, which is similar to that of Nishimoto & Gallant, but includes an additional V1 opponency stage.

The motion circuity described by Baker and Bair [13] included a population of units tuned to different motion directions. However, their population did not span multiple motion speeds or texture frequencies. To make the model respond realistically to a wider range of stimuli, I replaced their groups of twelve direction-selective units with the same filter bank that I used for the Nishimoto and Gallant [148] model (1296 filters). A separate filter bank was used for each eye (2592 filters in total). I used the same procedure to find the optimal weights as I did for Nishimoto and Gallant [148] model. More specifically, given the normalized responses of spatiotemporal filters corresponding to the left and right

eye $\mathbf{X}^l_{train}$ and $\mathbf{X}^r_{train}$ shown the same training movie (zero disparity), I first calculated the motion-opponent suppressed responses in each eye $\mathbf{O}^l_{train}$ and $\mathbf{O}^r_{train}$. For example for the left eye,

$$\mathbf{O}^l_{train} = \left[\mathbf{X}^l_{train} - c_{opp}\mathbf{Y}^r_{train}\right]_+, \tag{4.19}$$

where $\mathbf{Y}^r_{train} \in \mathbb{R}^{2000\mathrm{M}\times\mathrm{N}}$ is the result of reordering $\mathbf{X}^r_{train}$ such that each column corresponding to a filter's response with direction θ was replaced by the column corresponding to the opponent filter (i.e., a filter with $\theta - 180°$ direction) and c_{opp} is the motion-opponency parameter (e.g., $c_{opp} = 0.5$ means the normalized V1 responses from the opponent motion filters are scaled by 0.5 before being subtracted). Finally, $[]_+$ denotes half-wave rectification.

I then calculated the binocular-integrated response in the left and right eye, $\mathbf{M}^l_{train}$ and $\mathbf{M}^r_{train}$. For example for the left eye,

$$\mathbf{M}^l_{train} = b\,\mathbf{O}^l_{train} + (1-b)\,\mathbf{O}^r_{train}, \tag{4.20}$$

where b is the binocular-integration parameter. I set $b = 0.5$.

I defined the error function,

$$E(\mathbf{w}) = \|\mathbf{P}_{train}\mathbf{w} - \mathbf{R}\|^2 + \lambda\|\mathbf{w}\|^2, \tag{4.21}$$

where $\mathbf{w} \in \mathbb{R}^{2\mathrm{N}}$ is the weight vector, $\mathbf{P}_{train} \in \mathbb{R}^{2000\mathrm{M}\times 2\mathrm{N}}$ is a matrix containing the concatenated binocular-integrated responses $\mathbf{O}^l_{train}$ and $\mathbf{O}^r_{train}$ when the training movie was used as input, $\mathbf{R} \in \mathbb{R}^{2000\mathrm{M}}$ is a vector containing the target MT responses after transforming by the inverse of nonlinearity $a\exp(bx)$, and λ is a regularization constant. I finally found the weights using regularized linear regression, as

$$\mathbf{w} = \left(\mathbf{P}^T_{train}\mathbf{P}_{train} + \lambda I\right)^{-1}\mathbf{P}^T_{train}\mathbf{R}, \tag{4.22}$$

where T denotes matrix transpose, -1 denotes matrix inverse, and I denotes the identity matrix.

After finding the weights, the predicted MT responses to the test movie was calculated as,

$$\mathbf{mt} = a \exp(b \mathbf{P}_{test} \mathbf{w}), \tag{4.23}$$

where exp() denotes the exponential function, and a and b are the parameters of this nonlinear function.

4.2.7 Prediction of Unseen MT Data

I validated the empirical model by predicting a neural dataset that had not been used to develop or parameterize the model. Specifically, I used 73 speed-tuning curves from a previous study where MT cells were shown patches of random-dot stimuli moving in eight different motion speeds [27]. I created model neural populations of different sizes, and found how well the single most-similar model neuron accounted for the response of each MT cell. The inputs to the model were random-dot stimuli that were based on the description in [27].

I also used this dataset to validate and test sensitivity to a related response distribution parameter (see Section 4.3.3), specifically the scale parameter of the gamma distribution from which I drew the speed-tuning widths (see Table 4.1). I compared how well the proposed model predicted the speed-tuning dataset with my original scale parameter versus a range of alternative scales.

4.3 Results

4.3.1 Tuning Curve Approximation Examples

I tested how accurately the proposed model could reproduce tuning curves of real MT neurons from the electrophysiology literature. For each tuning curve, I generated the same kinds of visual stimuli (e.g., drifting gratings, plaids, and fields of moving random dots)

that were shown to the monkeys. I used these stimuli as input to the model, and optimized the model parameters to best fit the neural data.

Table 4.2 summarizes the results of the tuning curve fits for the proposed model, which I call Lucas-Kanade Nonlinear-Linear-Nonlinear (LKNLN), and my adaptations of the previous models by Nishimoto and Gallant [148] (NG) and Baker and Bair [13] (BB). Note that Baker and Bair [13] provide a software implementation of their model, but it has a small filter bank (see Methods) that is inadequate for processing many stimuli. I optimized relevant model parameters individually for each tuning curve. In my LKNLN model, there are relatively few such parameters, because the tuning curves are independent, and I did not change the calculation of the input fields. So only the parameters of the relevant tuning function and final nonlinearity were optimized. For the NG and BB models I optimized all the models' variable parameters, including weights of the spatiotemporal filters, for each tuning curve. Examples of tuning curve fits are shown in the following figures. Sources of error in my empirical model include non-ideal behaviour of the computer-vision methods operating on input images, and the data falling outside the tuning curve function family.

Figure 4.2 shows the speed tuning curves of four neurons (with different preferred speeds) where the monkeys were shown fields of random dots moving with different speeds. The proposed model approximates the neural data more closely than my adaptations of the models of Nishimoto and Gallant [148] and Baker and Bair [13].

Figure 4.3 illustrates the speed tuning of a neuron for moving random dots in two cases: when dot luminance was high, resulting a high contrast stimulus (Figure 4.3A), and when dot luminance was low, resulting a low contrast stimulus (Figure 4.3B). As shown in the figure, increasing the contrast not only modulated the response gain (peak spike rate) but it also shifted the preferred speed (position of the peak on the speed axis). The proposed model reproduces both these phenomena, whereas the previous models reproduce only the first. Note however that my empirical model does not provide a mechanistic explanation of the MT data, but only a fit.

Figure 4.4 shows models' fits to data on the effect of attending to stimuli in a neuron's

	#Tuning Curves	LKNLN	NG	BB
Speed	11 (8)	0.0531	0.1075	0.1654
Speed/Contrast	2 (8)	0.0543	0.2959	0.3650
Attention/Direction	2 (12)	0.0384	0.0848	0.1100
3D Motion	8 (12)	0.2144	NA	0.2450
Stimulus size	2 (7)	0.0667	0.0841	0.0599

Table 4.2: Summary of RMSE comparison between the proposed model (LKNLN), Nishimoto and Gallant [148] (NG), and Baker and Bair [13] (BB) to the neural data for different tuning parameters. The second column provides the number of tuning curves (along with the number of points in each tuning curve). Note that the NG model is monocular, so it does not reproduce binocular phenomena.

receptive field. Attending to stimuli modulates responses of different MT neurons to varying degrees. While the proposed model received attention masks, there was no mechanism for attention modulation in the other models. In my adaptations of these models, I modulated their responses with a scalar gain for attended stimuli. This gain was found such that the mean-squared error of data and model responses were minimized.

Majaj et al. [125] showed that motion integration by MT neurons occurs locally within small sub-regions of their receptive fields, rather than globally across the full receptive fields. They identified two regions within the receptive fields of a neuron where presenting the stimulus evoked similar neural responses. Then, they studied motion integration by comparing the direction selectivity of MT neurons to overlapping and non-overlapping gratings presented within the receptive field. Since motion integration was local, the ability of the neurons to integrate the motions of the two gratings was compromised when gratings were separated. The proposed model approximates this neural behaviour well (see Figure 4.5). According to Nishimoto and Gallant [148], their model does not account for this phenomenon, and extending it to do so would require including nonlinear interactions between the V1 filters of the model, which would drastically increase the number of parameters, making estimation more difficult. Other previous models that treat overlapping and non-overlapping features identically [193, 180, 13] would also not reproduce this phenomenon.

MT neurons also encode binocular disparity, with a variety of responses across the MT population, including preferences for near and far disparities, and various selectivities and depths of modulation. The proposed model closely approximates a wide variety of MT neuron disparity-tuning curves (Figure 4.6).

Recent studies [51] have revealed that some MT neurons respond to 3D motion, confirming area MT's role in encoding information about motion in depth. Figure 4.7 shows the neural responses of two different neurons to monocular and binocular stimuli. One neuron (Figure 4.7A-D) is tuned for fronto-parallel motion while the other neuron is tuned for motion toward the observer (Figure 4.7E-H). The proposed model approximates both

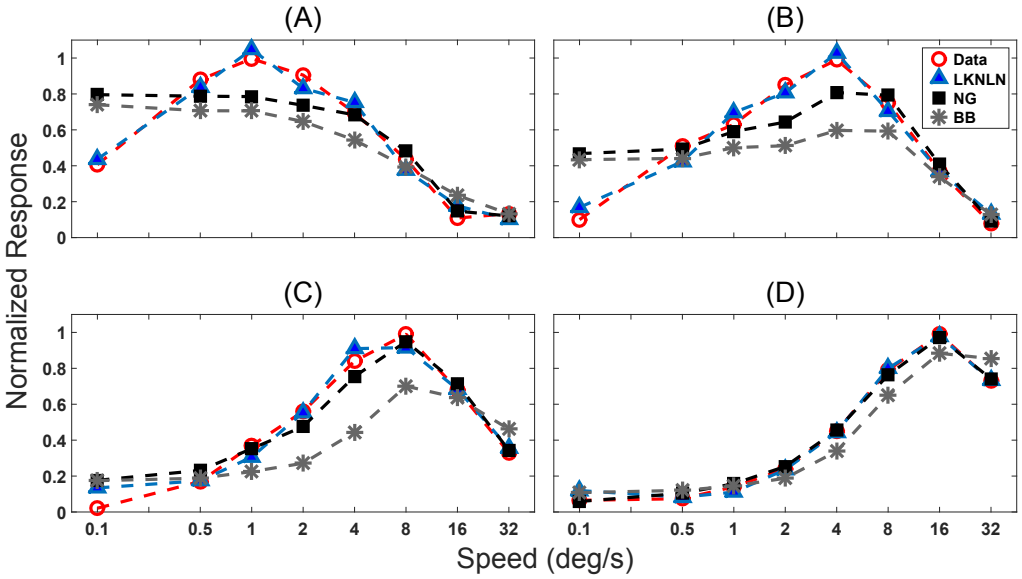


Figure 4.2: Speed tuning curves of four MT neurons, plotted on a logarithmic speed axis. Responses have been normalized so that the peak response of each neuron is equal to 1. Mean ± SD error for (A): 0.00 ± 0.06 spike/s (LKNLN), 0.00 ± 0.18 spike/s (NG), and 0.06 ± 0.21 spike/s (BB); for (B): -0.01 ± 0.06 spike/s (LKNLN), -0.00 ± 0.18 spike/s (NG), and 0.09 ± 0.23 spike/s (BB); for (C): -0.01 ± 0.06 spike/s (LKNLN), -0.00 ± 0.08 spike/s (NG), and 0.11 ± 0.21 spike/s (BB); for (D): -0.00 ± 0.02 spike/s (LKNLN), -0.00 ± 0.02 spike/s (NG), and 0.02 ± 0.09 spike/s (BB). Data replotted from Nover et al. [149].

types of neuron.

Size tuning is a result of antagonistic surrounds. Increasing the size of the stimulus to a certain point (optimal size) will increase an MT neuron's response, while larger-than-optimal stimuli evoke smaller responses. Figure 4.8 shows an approximation of two size-tuning curves using a symmetric difference-of-Gaussians kernel, one of three types that I adapt from [223].

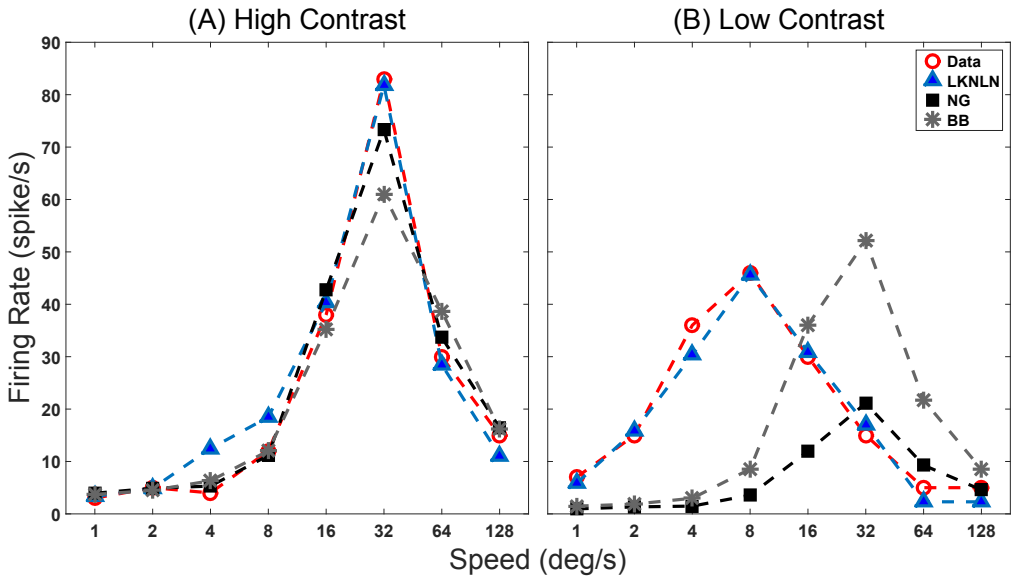


Figure 4.3: Effect of contrast on speed tuning curves. A, Speed tuning in high contrast. Mean ± SD error: -1.19 ± 3.35 spike/s (LKNLN), -0.18 ± 4.40 spike/s (NG), and 1.55 ± 8.90 spike/s (BB). B, Speed tuning in low contrast. Mean ± SD error: 1.19 ± 2.97 spike/s (LKNLN), 13.09 ± 17.83 spike/s (NG), and 3.22 ± 24.84 spike/s (BB). Contrast modulates the response and also shifts the peak (i.e., the preferred speed). While contrast modulates the response amplitude in all three models, only the proposed model (LKNLN) accurately shifts the peak. Data replotted from Pack et al. [156].

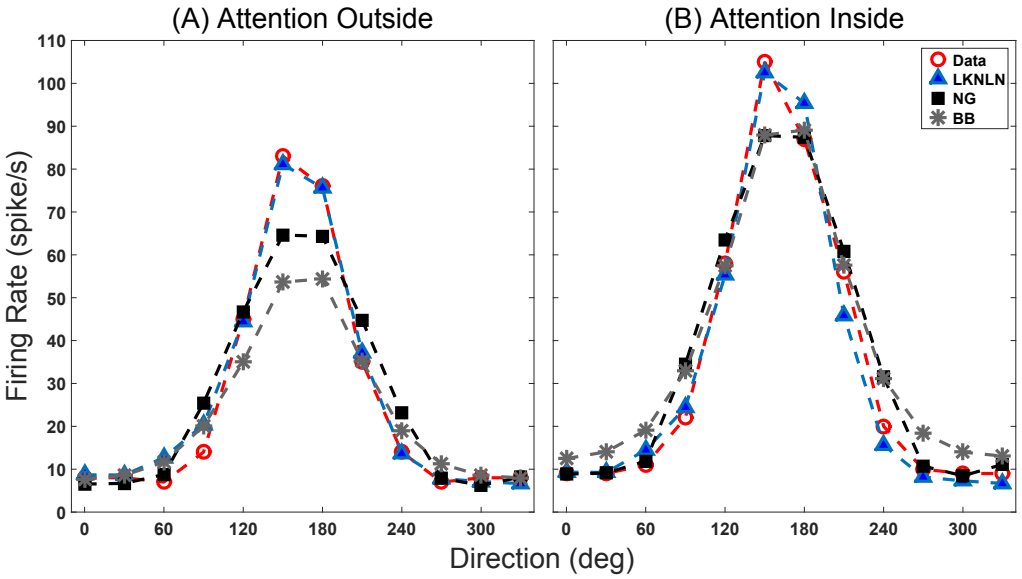


Figure 4.4: Attentional modulation of direction tuning. A, When the stimulus inside the RF was not attended. Mean ± SD error: -0.90 ± 2.73 spike/s (LKNLN), -0.02 ± 8.51 spike/s (NG), and 3.34 ± 11.26 spike/s (BB). B, When the stimulus inside the RF was attended. Mean ± SD error: 0.90 ± 4.63 spike/s (LKNLN), -1.73 ± 7.44 spike/s (NG), and -3.52 ± 7.43 spike/s (BB). Neural data for both cases replotted from Treue and Martínez Trujillo [211]. The proposed model (i.e., LKNLN) receives attention masks as input, so I defined the masks so that they did not cover the stimulus for the unattended case and covered for the attended case. For the other two models, I first found the best fit for the unattended case by multivariate regression. Given the unattended solution I then found the gain that minimized the error difference between the attended tuning curve and the modulated unattended solution.

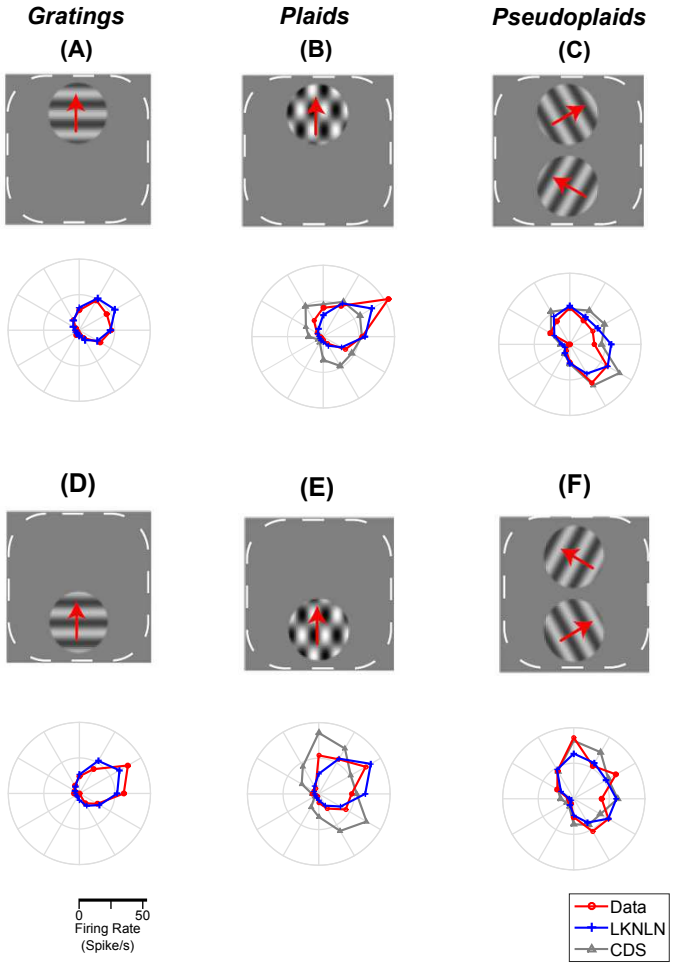


Figure 4.5: Response of an MT cell to gratings and plaids placed within different regions of the cell's receptive field (RF). The response magnitude is plotted on the radial axis, and the angular axis is the direction of motion. A,D, The neuron's response to grating stimuli at two different patches within RF. B, E, The neuron's response to plaids placed at two different regions over RF. The plaid stimuli are made by overlapping two gratings oriented 120° apart. Since this cell is selective for the motion of plaids independent of the orientation of their components (gratings), it is classified as a pattern direction selective (PDS) neuron. D, F, The two grating components of the plaids in (B,E) separated to different parts of the receptive field. If motion integration in MT cells were global (i.e., if these cells simply pooled all of their inputs from V1 cells), these plots would be similar plots as (B,E). Instead, the response in this case is close to the component direction selective (CDS) prediction, indicating that motion integration in MT cells are local rather than global. The proposed model produces realistic responses. Neural data (red) and CDS prediction (gray) replotted from Majaj et al. [125]; blue is the proposed model.

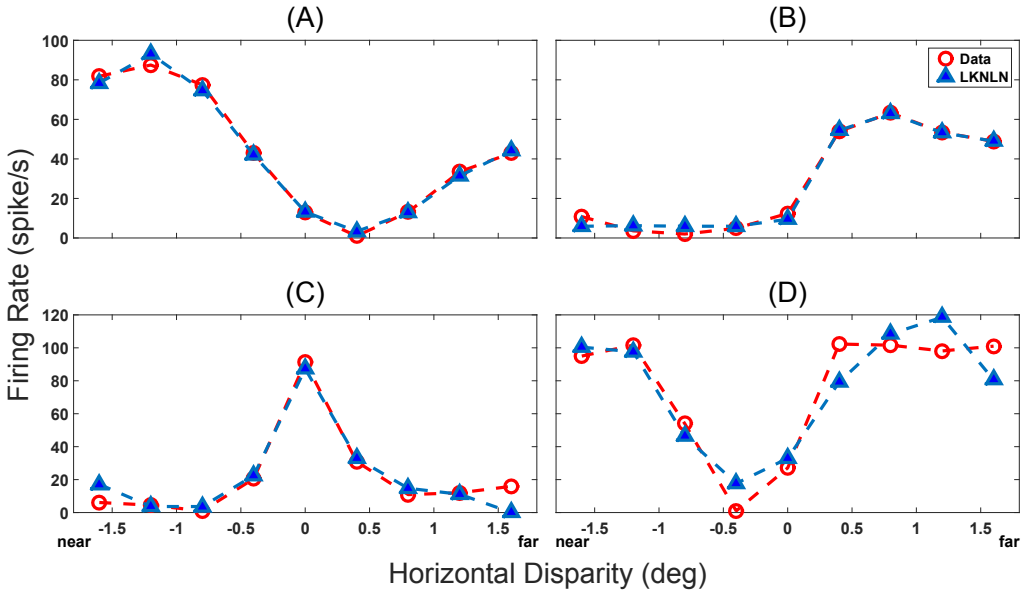


Figure 4.6: Disparity tuning curves of four neurons. Data replotted from [56]. A, Near (0.00 ± 1.96 spikes/s; mean error $\pm$ SD). B, Far (0.50 ± 1.58 spikes/s; mean error $\pm$ SD). C, Tuned-zero (-0.43 ± 2.93 spikes/s; mean error $\pm$ SD). D, Tuned inhibitory (1.38 ± 3.77 spikes/s; mean error $\pm$ SD).

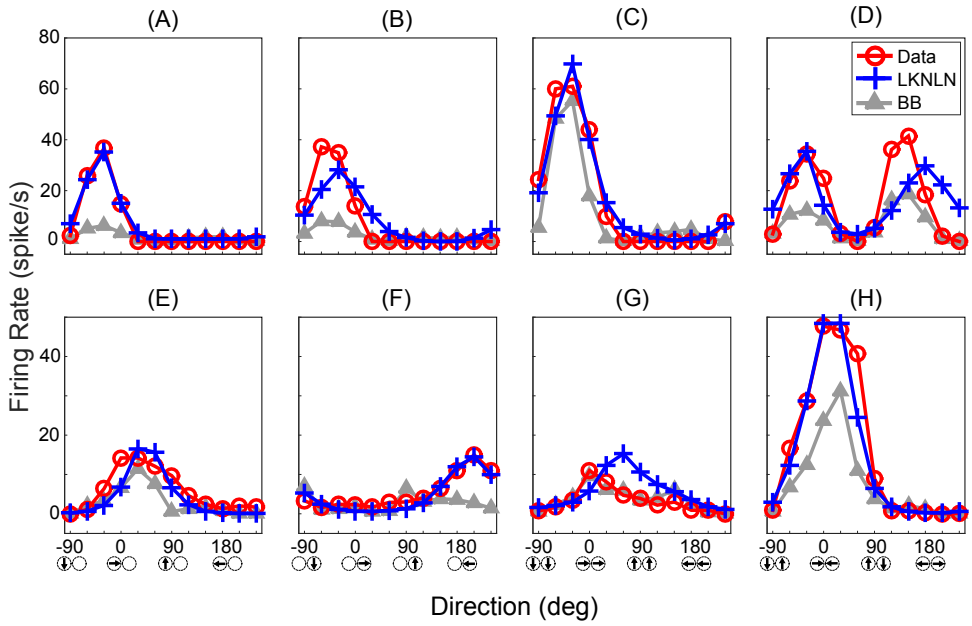


Figure 4.7: Examples of direction tuning of two MT cells to monocular and binocular stimuli. A–D, An MT neuron tuned for frontoparallel motion. A-B, Direction tuning for gratings presented monocularly to the left (A) and right eye (B). C, Direction tuning for binocular presentation of identical gratings. D, Direction tuning for gratings drifting in opposite directions in the two eyes. E–H, Responses of an MT neuron tuned for motion toward the observer. Direction tuning curves for monocular gratings (E, F), binocularly matched (G), and binocularly opposite motion (H). Neural data replotted from Czuba et al. [51] in red, prediction of the proposed model (LKNLN) in blue, and prediction of Baker and Bair [13] model (BB) in gray. Mean ± SD error, A: -1.11 ± 1.77 spikes/s (LKNLN) and 4.62 ± 10.63 spikes/s (BB), B: -0.25 ± 7.04 spikes/s (LKNLN) and 5.70 ± 11.39 spikes/s (BB), C: -0.61 ± 5.25 spikes/s (LKNLN) and 5.27 ± 9.85 spikes/s (BB), D: -0.71 ± 12.91 spikes/s (LKNLN) and 8.95 ± 9.40 spikes/s (BB), E: 1.48 ± 2.88 spikes/s (LKNLN) and 2.84 ± 3.02 spikes/s (BB), F: 0.52 ± 1.30 spikes/s (LKNLN) and 2.58 ± 4.84 spikes/s (BB), G: -2.47± 3.98 spikes/s (LKNLN) and -0.45 ± 1.42 spikes/s (BB), H: 1.42 ± 4.96 spikes/s, (LKNLN) and 8.15 ± 10.85 spikes/s (BB).

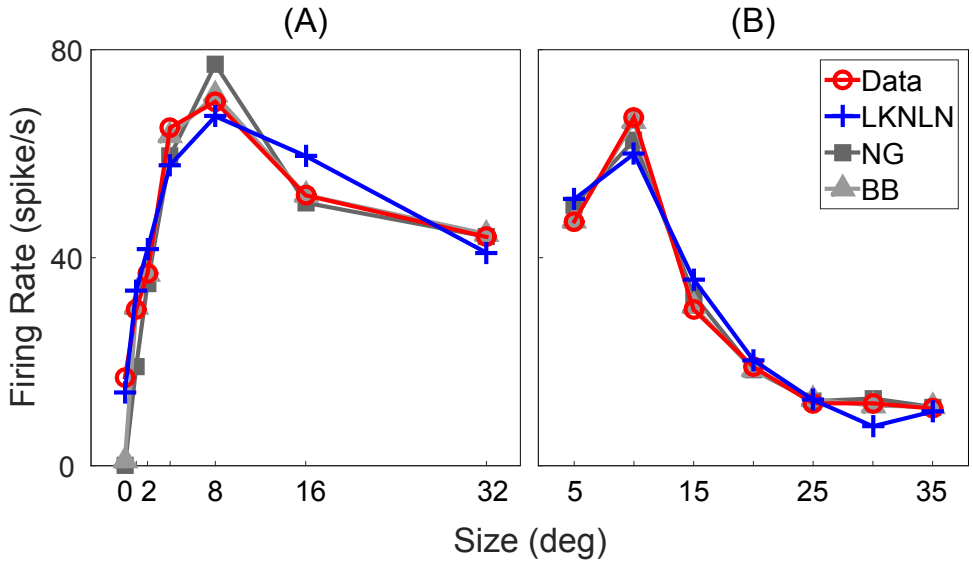


Figure 4.8: Two examples of size tuning curves. The kernels, which gave rise to the size tuning in the proposed model (LKNLN), were radially symmetric difference of Gaussians centred at the centre of video frames (the same as neuron's receptive field centre). A, Neural data replotted from DeAngelis and Uka [56]. Mean ± SD error: -0.00 ± 5.32 spikes/s (LKNLN), 4.21 ± 7.86 spikes/s (NG), and 2.18 ± 6.16 spikes/s (BB). B, Neural data replotted from Pack et al. [156]. Mean ± SD error: 0.00 ± 4.54 spikes/s (LKNLN), -0.35 ± 2.50 spikes/s (NG), and -0.01 ± 0.59 spikes/s (BB).

4.3.2 Dynamics of Pattern and Component Selectivity

Figure 4.9 shows the distribution and dynamics of pattern and motion selectivity in the empirical model. The model closely approximates the data from Smith et al. [197].

As described in the Methods, I experimented with four different ways of combining pattern and component responses. To compare performance between these different forms, I used the population Pearson correlation coefficient between Z-scores that I randomly drew from the distributions, which were approximated for each time window, and the Z-scores that I calculated after building the response S_t based on S_c, S_p, and $\boldsymbol{p}$, which I found in the optimization process. Table 4.3 summarizes the results for a population of 500 neurons. The best results were obtained by the compressive form where I linearly combined pattern response, component response, and a third term, which was constructed by passing the sum of these two responses through a compressive nonlinearity.

4.3.3 Parameter Distributions

The empirical model is meant to closely approximate population activity in MT, so statistical distributions of parameters are also an important part of the model. Such distributions have frequently been estimated in the literature. However, past computational models of MT have typically not attempted to produce realistic population responses, except along a small number of tuning dimensions [e.g., 149].

Figure 4.10 shows nine examples of fits of parameter distributions. In each case I chose the best of seven different distributions according to the Akaike Information Criterion [3], as described in the Methods.

4.3.4 Neural Response Predictions

Beyond examining fits of published tuning curves and distributions of response properties, I further validated the model using a more detailed dataset of speed tuning in 73 MT

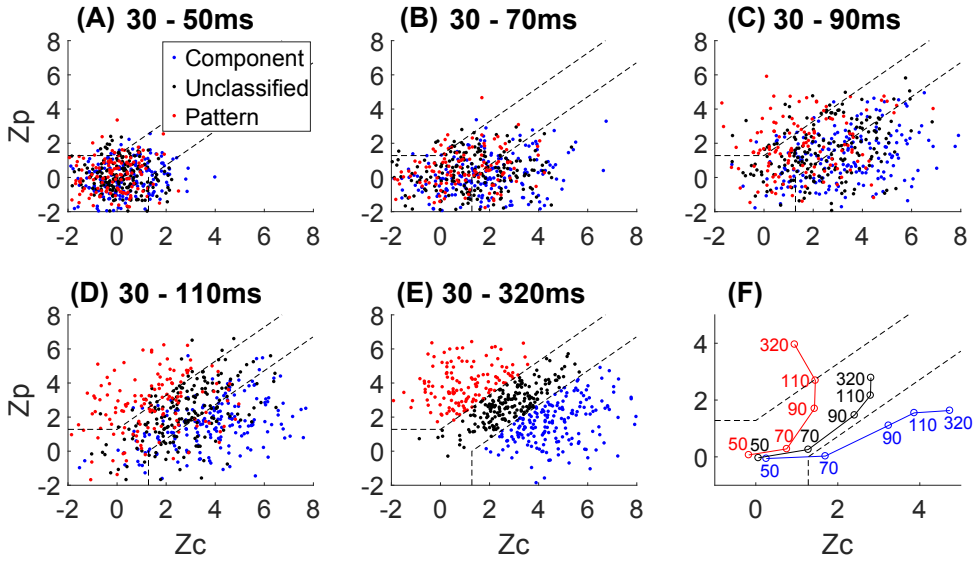


Figure 4.9: Pattern selectivity of empirical model. A-E, Scatterplots of Z-transformed pattern and component correlations (Z_p and Z_c) for 500 modelled neurons over time. The red and blue dots represent the pattern and component neurons, respectively. The black dots represent neurons which are not classified. For the final time window (E), I classified each neuron based on its location on the Z-transformed-correlations plane as in Smith et al. [197]. For other time windows (A-D), I used Hungarian algorithm to match each sample in a time window (e.g., D) to its latter time window (e.g., E) so that the total Euclidean distance between matched samples, perturbed with Gaussian noise (0 +/- 2.5SD), was minimized. F, the time evolution of each class. Each data point represents the average Z_p and Z_c values, of a particular class, in a time window whose ending time has been written next to it (see Figures 5-6 of Smith et al. [197] for comparison with actual neural data; I do not replot the data here because some of the dots are too dense to be extracted accurately).

Form	30-50ms	30-70ms	30-90ms	30-110ms	30-320ms
Additive	0.31	0.52	0.06	0.57	0.53
Multiplicative	0.65	0.98	1.00	0.993	0.80
Expansive	0.99	0.99	0.99	1.00	0.91
Compressive	1.00	1.00	1.00	1.00	0.95

Table 4.3: Summary of comparison between four different forms of combining component and pattern direction selective responses. A population of 500 neurons was modelled. Numbers indicate the population Pearson correlation coefficients between the sampled and calculated Z-scores based on a specific form for the corresponding time window. For example, 0.53 in the last column of the second row indicates that $\rho_{sampled,calculated} = 0.53$ where *sampled* refers to the population of 1000 sampled Z-scores (500 Z_cs and 500 Z_ps) drawn from the modelled distribution of 30-320ms time window, and *calculated* means the Z-scores calculated for S_c, S_p, and S_t where S_t was calculated by combining S_c and S_p signals in the additive form (see Equation 4.16). The compressive form had the best performance (highest Pearson correlation) in all time windows.

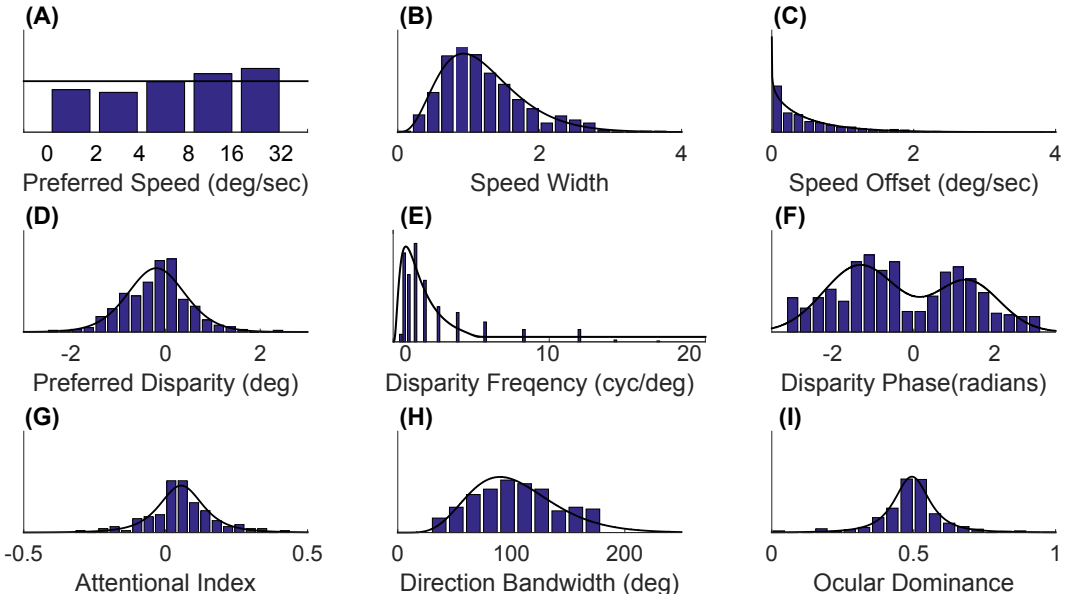


Figure 4.10: Examples of parameter distributions. In each case I replot the data (histograms) along with the selected distribution. A-C, speed parameters including preferred speed (log uniform) in logarithmic space, speed width (gamma), and speed offset (gamma) [149]. D-F, disparity parameters including preferred disparity (t location-scale), disparity frequency (log normal), and disparity phase (Gaussian mixture) [56]. G, Attentional index (t location-scale) [211]. H, Direction bandwidth (gamma) [220]. I, Ocular dominance (t location-scale) [56].

cells, from a previous study [27]. This experiment involved random dot stimuli moving coherently at one of eight different speeds (0.5, 1, 2, 4, 8, 16, 32, 64deg/sec).

I approximated the responses of these MT cells by creating a population of N synthetic neurons of my empirical model. I chose N to be 8, 16, 32, 64, 128, 265, or 1048. At each speed of motion, I recreated ten sequences of moving random dot stimuli (with the same dot size, density, contrast, and replotting scheme as the original study) and fed them to the synthetic neural population. The final response of each synthetic neuron in the population at each speed was calculated as the average of the ten sequences at that speed. Next, for each MT cell, I selected the synthetic neuron from the population that had the highest correlation with that MT cell. I calculated the coefficient of determination (r^2) as the proportion of the variance in the MT cell, which was predictable from that synthetic neuron. In summary, I used a nearest-neighbour approximation of each cell rather than linear-regression from the full model population.

Because of the stochastic population parameters of the empirical model, two N-neuron populations sampled from these distributions will not have identical responses. Therefore, instead of a single N-neuron population, I created five populations, repeating the above process for each population.

Figure 4.11 illustrates how the average explained variance for 73 MT cells increases as the empirical model grows in size. Each point of the curve is the average of 365 (73×5) r^2 values, because there were 73 MT cells and five different populations for any given population size.

I repeated this process with various scale parameters of the gamma distribution (see Table 4.1) from which the speed-tuning widths (see Equation 4.8) were drawn, to validate this parameter and test sensitivity to it. I chose 64 as the population size N and again created five different populations. As can be seen in Figure 4.12, there is a modest dependence on this parameter, and the averaged explained variance is indeed highest when the speed-tuning widths were drawn from the original estimate indicating my accurate estimation of this parameter.

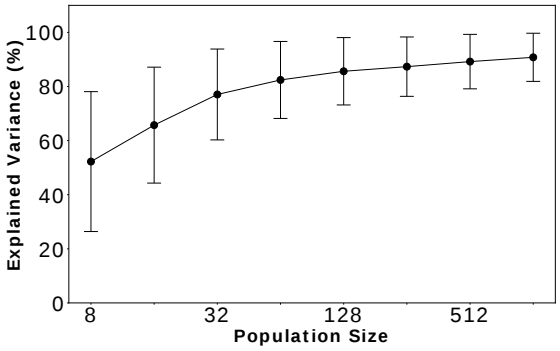


Figure 4.11: Explained variance vs. population size of empirical model. As the population of the empirical model grows, the probability of having a synthetic neuron with more similar response increases. Each point and error bar, respectively, represents the average and standard deviation of 365 (73×5) r^2 values (73 MT cells times five different model populations for any given population size).

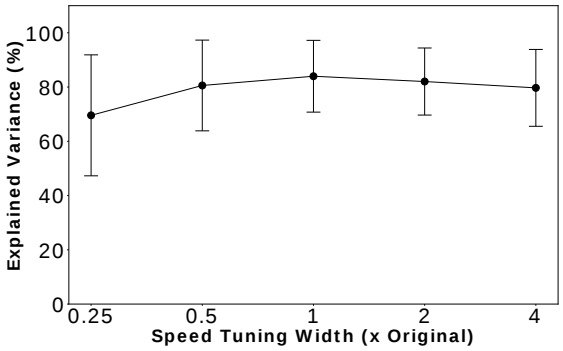


Figure 4.12: Explained variance vs. the scale parameter of the speed-tuning-width gamma distribution of empirical model (see text for details). I changed the scale parameter by multiplying it with one of [0.25, 0.5, 1, 2, 4] values. The model population size was 64. Each point and error bar, respectively, represents the average and standard deviation of 365 (73×5) r^2 values. The original scale parameter produced the best predictions on average.

4.4 Discussion

I developed a video-driven, empirical model of activity in the primate middle temporal area (MT) that emulates many tuning properties and statistics from the literature. The model uses well-supported tuning curves, and well-established computer-vision methods of generating represented signals such as speed and disparity.

As far as I know, this is the most thorough video-driven model of MT population activity developed so far. I expect that it will be useful in the future for examining relationships between features of MT population activity and performance of tasks that make use of visual motion information.

Compared with other MT models [162, 216], a limitation of my approach is that its responses are not produced by biologically plausible mechanisms. That is, the model is empirical rather than mechanistic. This may impair the model's ability to generalize beyond the source data. This limitation might be mitigated if the model is used to guide representations in more mechanistic models, such as convolutional networks (see Chapter 7), or perhaps more physiologically detailed deep networks.

Chapter 5

Sensitivity Analysis of MT Parameters on Visual Odometry Task

5.1 Introduction

The empirical MT model provides the possibility of investigating the influence of individual MT response properties on task performance. More specifically, it is possible to manipulate tuning statistics by changing the parameters of the model to closely explore the relationships between representations and behaviour. In this chapter, I investigate how changing two MT tuning features affect the visual odometry task, which is the process of estimating self-motion from video.

For this investigation, I designed a convolutional neural network (CNN) that received MT activity (i.e., output of the empirical model to video) and predicted velocities corresponding to the motion of a synthetic camera system within a virtual world. The data (i.e., video and ground-truth velocities) came from a novel synthetic odometry dataset, which I created and used to train and test the network.

The two selected features of MT response were (1) direction-tuning bandwidth and (2)

speed-tuning width. A large percent of MT neurons are sensitive to both direction and speed. The tuning for direction and speed are highly relevant properties of MT code for solving odometry. Finding how performance would be affected by manipulating direction and speed-tuning widths was particularly valuable since the issue of optimal neuronal tuning widths has received much attention in the literature [33]. Theoretically, arguments have been made for both sharp [16, 232] and broad [14, 67, 89, 73] tuning curves as a means to increase encoding accuracy. However, combining the MT model with deep networks created an experimental setting where I could look for the optimal tuning widths for solving a realistic task.

Next, I give a more detailed definition of the odometry task, which is a key topic in robotics. I also talk briefly on how the brain solves odometry with a focus on the role of visual information.

5.1.1 Visual Odometry

The word odometry is a contraction of two Greek words: *odos* (meaning "route") and *metron* (meaning "measure"). In robotics, odometry refers to the process of estimating change in position over time (with respect to a known initial position) by measuring egomotion of an agent (i.e., vehicle or robot). Odometry is an essential part of a SLAM system. A simultaneous localization and mapping (SLAM) system addresses the problem of a robot navigating an unknown environment. While navigating the environment, the robot seeks to acquire a map thereof, and at the same time it wishes to localize itself using its map [208].

The traditional approach for solving odometry is wheel odometry. Wheel odometry refers to estimating egomotion by integrating the number of turns of a robot's wheels over time. Since the accuracy of wheel odometry is prone to suffer from wheel slip, especially in uneven terrain, alternative approaches have been proposed based on acquiring information from different types of sensory systems. One good example of these approaches is visual odometry.

In visual odometry (VO), the information, which is used to estimate egomotion of a robot, comes solely from the image feed of a single or multiple cameras attached to the agent and no other sensory input is used. More specifically, VO operates by incrementally estimating the pose of the vehicle through examination of the changes that motion induces on the images of its onboard camera(s). For VO to work effectively, sufficient illumination in the environment and a scene with enough texture, to allow apparent motion to be extracted, are vital. Also, frame rate should be high enough so that the consecutive frames have sufficient scene overlap.

An extensive literature exists on VO where various methods, based on different approaches, have been proposed and compared (see Aqel et al. [9] for a recent review). Two categories of these approaches are feature-based and flow-based methods. The feature-based approaches involve extracting image features (e.g., lines, curves, and corners) between sequential image frames, matching or tracking the distinctive ones among the extracted features [9]. The flow-based approaches, on the other hand, first calculate the optic flow field for each frame and then do further analysis (consists of fitting to a sum of translation and rotation templates) to estimate camera egomotion corresponding to the flow field. The resulting sets of equations are often both highly over-determined and subject to ill conditioned inputs [37]. Therefore, statistical methods such as least-median-of-squares or RANSAC have been proposed to help screen out outliers and segment flow fields [11]

5.1.2 VO in Primate Brain

Primates use different sensory inputs (e.g., visual, auditory, and vestibular) to localize where they are during navigation. Among these sensory inputs, visual information plays the major role; so it is reasonable to expect the primate brain to implicitly solve the VO problem. Interestingly, the primate brain seems to apply two different strategies to solving this problem, each being conducted via a different brain circuitry (i.e., the dorsal and ventral streams). The computation in the ventral stream is based on recognizing land marks and features of the scene (like feature-based methods). The dorsal stream, on the

other hand, encodes motion information to help the animal localize itself (like flow-based methods).

MT is a key visual area of the dorsal stream, which encodes motion information (see Section 2.2). Therefore, my approach of combing the MT model and deep networks for solving odometry can be categorized as a flow-based method.

5.2 Methods

I chose direction-tuning bandwidth and speed-tuning width for sensitivity analysis. In the MT model, both of these parameters were drawn from gamma distributions (see Table 4.1). A gamma distribution has two parameters, shape and scale. I altered each of these two gamma distributions by changing their scale parameters. Hence, the distribution of speed-tuning widths and direction-tuning bandwidths changed and I could examine how these changes influenced odometry performance. I also simplified the MT model by removing the dynamics of pattern and component selectivity and used difference of Gaussian kernels as receptive fields.

5.2.1 A Novel Visual Odometry Dataset

I needed a stereo odometry dataset with high frame rate, which had enough number of frames (i.e., at least a few hundred thousand) with ground-truth trajectories for training deep networks. The existing datasets for visual odometry include KITTI with 22 stereo sequences (http://www.cvlibs.net/datasets/kitti/eval_odometry.php), Monocular Visual Odometry Dataset (https://vision.in.tum.de/data/datasets/mono-dataset), monocular RGB-D SLAM Dataset (https://vision.in.tum.de/data/datasets/rgbd-dataset), and the Wean Hall dataset (http://www.cs.cmu.edu/~halismai/wean/). However, all these datasets lack one or several desired specifications. For example, many of them are monocular while KITTI do not have enough samples for training a deep network.

Figure 5.1: Two example stereo frames from novel odometry dataset. Both left and right frames were 76×76 pixels.

Not finding an existing dataset with desired requirements, I created a new synthetic dataset in Unreal Engine 4. I used the Modular Neighbourhood Pack, which contains a residential neighbourhood that looks fairly realistic. More specifically, the virtual neighbourhood contained houses, cars, and streets and it was surrounded by a natural landscape of grass and trees with different shapes, sizes, and textures.

I used UnrealCV plugin [166] to move a stereo camera system along curvilinear paths inside this virtual world. The baseline of the camera system was 60mm, which is within the range of human interpupillary-distance. The dataset consisted of "moves" of six frames each, starting at different locations and moving along different trajectories. For each move, I drew random medio-lateral, antero-posterior, and angular velocities, and collected six grayscale 76×76 stereo frames (at 60 FPS). Figure 5.1 depicts two example stereo frames from the dataset.

The dataset had 75000 moves for training and 9000 moves for validation and testing.

Layer	# Kernels	Kernel Size	Shape	Pool	Nonlinearity
Conv-1	128	9×9	6×6	None	ReLU
Conv-2	128	9×9	6×6	2×2	ReLU
Dense			1024		ReLU
Output			3		None

Table 5.1: Structure of the CNN used for sensitivity analysis of MT features on the visual odometry task.

I then calculated dense direction, speed, and disparity fields (pyramidal Lucas-Kanade method) as well as contrast fields (Peli method) for every frame of the sequence. For each of these four fields, I fed the sequence-average field to the MT model, to produce the input for the deep network. The input therefore reflects average stimulus features over several frames, roughly consistent with the low-pass properties of MT neurons [12].

Therefore, for each move in the dataset, the deep network was provided with MT responses as input and medio-lateral, antero-posterior, and angular velocities as regression targets.

5.2.2 Architecture of the CNN

I designed a convolutional network with two convolutional layers, one pooling layer, and two dense layers. This design was a simplified version of the deep network, which I discuss in Section 7.2.2, where the design was based roughly on the dorsal stream. The output layer of this network had three units to estimate medio-lateral, antero-posterior and angular velocities from the MT model. Table 5.1 summarizes the structure of this CNN.

5.2.3 Training

The network was trained using the mean-square error (MSE) loss. More formally,

$$E = \sum_{v} (y_v - t_v)^2, \tag{5.1}$$

where t_vs are target velocities and y_vs are network outputs.

Adam (with the default parameters) [107] was used as the optimizer algorithm. Batch Normalization (see Section 3.2.7) was also used after all layers of the network (except the output layer) to reduce overfitting and speed up training. The CNN was implemented in Keras [44] with TensorFlow [1] backend, and trained on an NVIDIA GeForce GTX Titan Xp GPU.

5.3 Results

Figure 5.2 illustrates the RMSE of the odometry task vs. different scale parameters for the gamma distribution of direction-tuning bandwidths. To change the bandwidth, I multiplied the original scale parameter, which I had estimated from literature, by each of $[0.25, 0.5, 1, 2, 4]$. For each neuron of the population, I then calculated its direction-tuning width (i.e., σ_θ in Equation 4.11) using the minimum of the drawn direction-tuning bandwidth and 360° [1]. The ∞ symbol refers to not having any direction selectivity in the model (i.e., direction bandwidth is infinite). I found the best performance at four times the original scale parameter. To verify this, I created another three populations, all using four times the original scale parameter. The average RMSE of these four populations was lower than other cases, although the 0.5x, 1x, and 2x means differed by less than five percent. I

[1] Note that adding an upper bound on the direction-tuning bandwidth meant that multiplying the scale parameter by eight or any larger number (e.g., 8x or 16x) would often result in populations with 360° for almost all neurons (as nearly all of the drawn values were larger than 360°). However in the 4x case, about 40% of the bandwidths were still smaller than 360° including a few with narrower than 120° bandwidths.

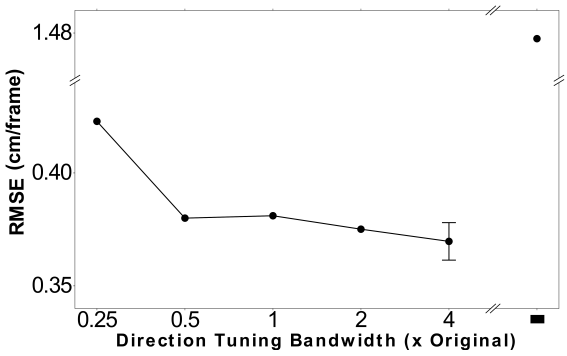


Figure 5.2: Task performance comparison with respect to changing direction-tuning-bandwidth distribution of the empirical model. To change the distribution I multiplied the original scale parameter of the modelled gamma distribution with $[0.25, 0.5, 1, 2, 4]$. The ∞ symbol refers to the case where I omitted direction selectivity of the response. For four times the original scale parameter case, I created four different populations (hence the error bar). The standard deviation of the target velocities of the validation set was 1.54 cm/frame.

tested statistical significance of differences in mean absolute errors with each of these scale factors, compared to the 4x scale factor, using multiple t-tests. Only the 0.5x errors were significantly higher ($\alpha < .05$) with a Bonferroni correction for multiple comparisons.

Figure 5.3 shows the RMSE of the odometry task vs. different scale parameters for the gamma distribution of speed-tuning widths (σ_s in Equation 4.8), where I applied the same idea for the speed-tuning widths as I did for the direction-tuning bandwidths. In this case, two times the original scale parameter out-performed the other cases in all four different populations that I tested. Mean absolute errors in the 2x case were significantly lower than all other cases ($\alpha < .05$), accounting for multiple comparisons. Comparing the RMSE of ∞-width speed tuning (Figure 5.3) to that of ∞-width direction tuning (Figure 5.2) reveals that elimination of direction tuning had a noticeably larger impact than elimination of speed tuning.

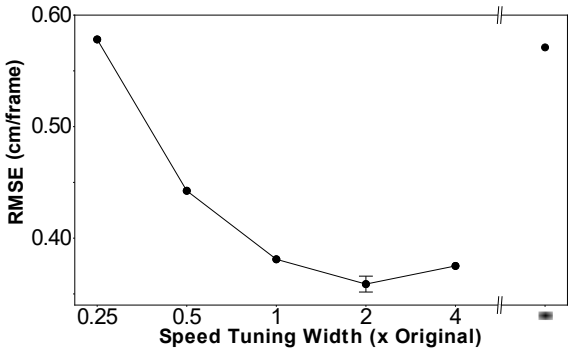


Figure 5.3: Task performance comparison with respect to changing speed-tuning-width distribution of the empirical model. To change the distribution I multiplied the original scale parameter of the modelled gamma distribution with $[0.25, 0.5, 1, 2, 4]$. The ∞ symbol refers to the case where I omitted speed selectivity of the response. The standard deviation of the target velocities of the validation set was 1.54 cm/frame.

5.4 Discussion

I this chapter, I presented a novel approach to study the relationships between MT response properties and task performance. As opposed to the traditional approaches (i.e., microstimulation and lesion studies) for finding causal links between activity of a brain area and a specific function, this approach essentially involves a model (e.g., the empirical MT model). However, this approach can reveal specific relationships between individual tuning features and task performance, something not possible with the traditional approaches.

Following this novel approach, I investigated the effects of modulating two MT tuning features on solving a visual odometry task. These were direction- and speed-tuning widths. The deep network, which was used for solving odometry, received MT activity and was retrained after each MT feature modulation. While the network generally learned to compensate for the moderate changes of the features, they still had a persistent effect on task performance. More specifically, the odometry task performance was more sensitive to moderate modulations of speed-tuning widths than to similar modulations of direction-

tuning widths. On the other hand, elimination of direction tuning had a more significant effect on odometry performance compared to elimination of speed tuning.

5.4.1 Future Work

While the optimal distribution parameters of the MT direction- and speed-tuning widths were found, more brain-like network architectures, normalization and regularization techniques should be used to validate these findings further. Overall, it will be more accurate if the same network is optimized to solve a range of tasks. In addition to visual odometry, visual tracking (which requires smooth-pursuit eye movement where MT is known to play an essential role [119]) or motion-based classification of behaviour are appropriate candidates for a more comprehensive study.

Another direction for sensitivity analysis is to remove deep networks and directly decode the target velocities from populations of synthetic neurons with different model parameters (e.g., different tuning widths) using the optimal linear decoding approach [185]. In that case, the reconstruction performance of the decoders can reflect a lower bound on the information that would be available for the downstream brain areas [58].

Chapter 6

Functional Role of Suppressive Surround of Area MT

6.1 Introduction

The receptive field (RF) of many visual neurons in different areas of the visual hierarchy such as the retina [136], V1 [86], and MT [167] has a centre-surround organization. More specifically, their RF consists of a central region (a.k.a. the classical receptive field) that is *surrounded* by another region, which does not elicit a response when stimulated alone but modulates neural activity when paired with a centre stimulus. This surround region (a.k.a. the extra-classical receptive field) is most commonly suppressive, which allows neurons to operate as differentiators, removing redundancy from the input and selectively encoding discontinuity across the input space [15].

The surround of MT neurons has been extensively studied. These studies investigated the MT surround direction and speed tuning [5], disparity tuning [29], contrast sensitivity [156, 98], and spatial structure [167, 223].

From the functional point of view, MT surround has been hypothesized to be useful

for figure–ground segregation [6] and the computation of three-dimensional shape from relative motion [34]. Additionally, the suppressive surround may play a role in redundancy reduction (by decorrelating neural responses [218]), input normalization [38], detection of edge discontinuities [195], and estimating heading direction (by improving the estimation of optic flow [49] as well as helping to identify the presence of moving objects during navigation [176]).

The source behind MT surround suppression is not well-known [49]. The fact that MT neurons in the cortical input layer IV lack suppressive surround indicates that surround inhibition observed in MT is not inherited from its feedforward inputs [204]. On the other hand, Liu et al. [122] found that local concentration manipulation of GABA or Gabazine, which acts as an antagonist at GABA receptors, did not affect surround suppression suggesting that local changes in the strength of inhibition by inhibitory interneurons has no effect on the surround suppression in MT cells.

Without concerning about the source or mechanism of the surround suppression, Cui et al. [49] developed a hierarchical MT model consisted of two suppression components (i.e., a direction-selective component and a non-direction-selective component) that were functionally relevant for studying MT activity in response to complex motion stimuli. More specifically, they used continuously varying optic-flow stimuli composed of moving dots whose velocity varied over space and time. This flow field was generated as a random combination of six optic flow components: (1) horizontal and (2) vertical translations, (3) expansion, (4) rotation, and (5) horizontal and (6) vertical shears. Their model had three parallel components. In addition to the two suppression components (corresponding to the surround), the model had an excitation component, which corresponded to the classical RF. Each component had subunits with separate speed and direction tuning functions, which were multiplied together to generate the response of the subunit (except the non-direction-selective surround component, which only had speed tuning). They used the von-Mises function as the direction tuning function and a linear combination of ten tent functions as the speed tuning function for each subunit. The responses of these subunits, at each component, were pooled using spatial weights (kennels), which were trainable parameters.

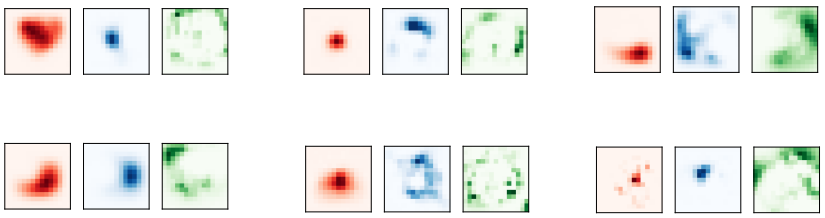


Figure 6.1: Six examples of the spatial kernels that Cui et al. [49] found to best fit neural data, reproduced from [49]. Each neuron has three kernels: excitation (red), direction-selective surround (blue), and non-direction-selective surround (green). Note that for a given neuron, the non-direction selective kernel is more dispersed than the direction selective kernel.

These weights were constrained to be non-negative for the excitation component and non-positive for the suppression components. The final response of the neuron was calculated by adding the responses of all three components, and passing the result through a static nonlinearity. Their model was more accurate in predicting neural responses compared to the motion-oppoency model, which had only the excitation component and no surround. They also found that for a given neuron the non-direction selective surround is more dispersed than the direction selective surround. Figure 6.1 illustrates six examples of the spatial kernels that they found to best fit neural data.

I adapted their modelling framework for MT into deep neural networks. Incorporating a modified version of the empirical MT model to these deep networks created a setting where I could study the functional role of MT surrounds in solving realistic tasks. These tasks were visual odometry and motion-based classification of different hand gestures, where both demanded complex-motion estimation.

6.2 Methods

6.2.1 Visual Odometry and Hand-Gesture Recognition Datasets

For each of the vision tasks, I employed a dataset on which I trained and tested the corresponding deep networks. For visual odometry the same dataset, which I created in Unreal Engine 4 (Section 5.2.1), was used. For hand-gesture recognition, I used the 20BN-JESTER dataset (https://20bn.com/datasets/jester). This dataset was a collection of about 150000 labeled video sequences with different resolutions and lengths, at 12 FPS. Each sequence showed a human performing a hand gesture from a predefinded list in front of a webcam. Examples of these hand gestures were thumbs up, shaking hand, swiping left or right, pushing hand away, and drumming fingers. Overall, there were 27 possible classes: 25 hand gestures, doing nothing, and doing other things.

For each video sequence in the dataset, I first resized the frames to 76×76 resolution, and then calculated flow, disparity, and contrast fields. Since the frame rate was already comparable to the temporal range of MT, unlike the higher frame rate of the odometry task, I did not feed the sequence-average of these fields as input. Instead, I chose a twelve-frame window from each sequence where the average flow was maximum compared to any other window. This procedure led to capturing the most motion-informative part of the sequence, while keeping the sequence small enough so a mini-batch could be fit on GPU memory during training of the deep CNN[1].

6.2.2 Structure of the MT Model

I modified the empirical MT model such that I could adapt the hierarchical framework that Cui et al. [49] suggested to model MT for studying MT surround. Figure 6.2 illustrates the architecture of my adaptation for an MT population of n modelled neurons. There are

[1]Note that increasing the sequence length beyond twelve frames up to fourteen frames (and consequently decreasing the mini-batch size) did not improve performance.

three parallel components (paths) in the model. The first component (red box) corresponds to the excitatory centre (i.e., classical receptive field) of MT cells. The second component (blue box) corresponds to the direction-selective suppressive surround. Finally, the third component (green box) corresponds to the non-direction-selective suppressive surround. This MT module was integrated into deep convolutional networks for solving vision tasks.

Note that by a population of n modelled neurons, I mean n sets of feature maps (model parameters) in each component regardless of the pixel dimensions of the feature map. So it should be understood that each feature map or model neuron is ultimately tiled across the visual field to simulate many neurons with different receptive field centres.

Figure 6.2 illustrates that each video input was used to calculate four fields (optic flow comprises two fields) with the same size as the input frames (i.e., field values are defined at each image pixel x, y). These fields were contrast, horizontal and vertical optic flow, and disparity. I used pyramidal Lucas-Kanade method (see Section 3.1.4) to calculate horizontal and vertical flow, as well as disparity fields. I also employed a local band-limited contrast calculation method, which I discussed in Section 3.1.5, to create a contrast field from video.

The next step was to calculate eight additional fields for each of n neurons (hence sets of n-channel feature maps in Figure 6.2); three fields for the excitation component, three fields for the direction-selective surround component, and two fields for the non-direction-selective component. I refer to these additional fields as the tuning fields. Each of these tuning fields was a point-wise nonlinear function of either contrast, flow, disparity, or a combination of them (see details in Section 4.2.3). For each component of a neuron, a final map was calculated as the point-wise product of the corresponding tuning fields (product maps). Next, I wanted to apply sparse convolutional kernels on the product maps such that each of the feature maps of the following layers (i.e., layers Exc, DS-Sup, NS-Sup in Figure 6.2) was connected, almost exclusively, to one of the product maps. Therefore, the ith feature map at the MT layer was the result of adding three individual maps (one from each of the Exc, DS-Sup, and NS-Sup layers), after the corresponding kernels were applied

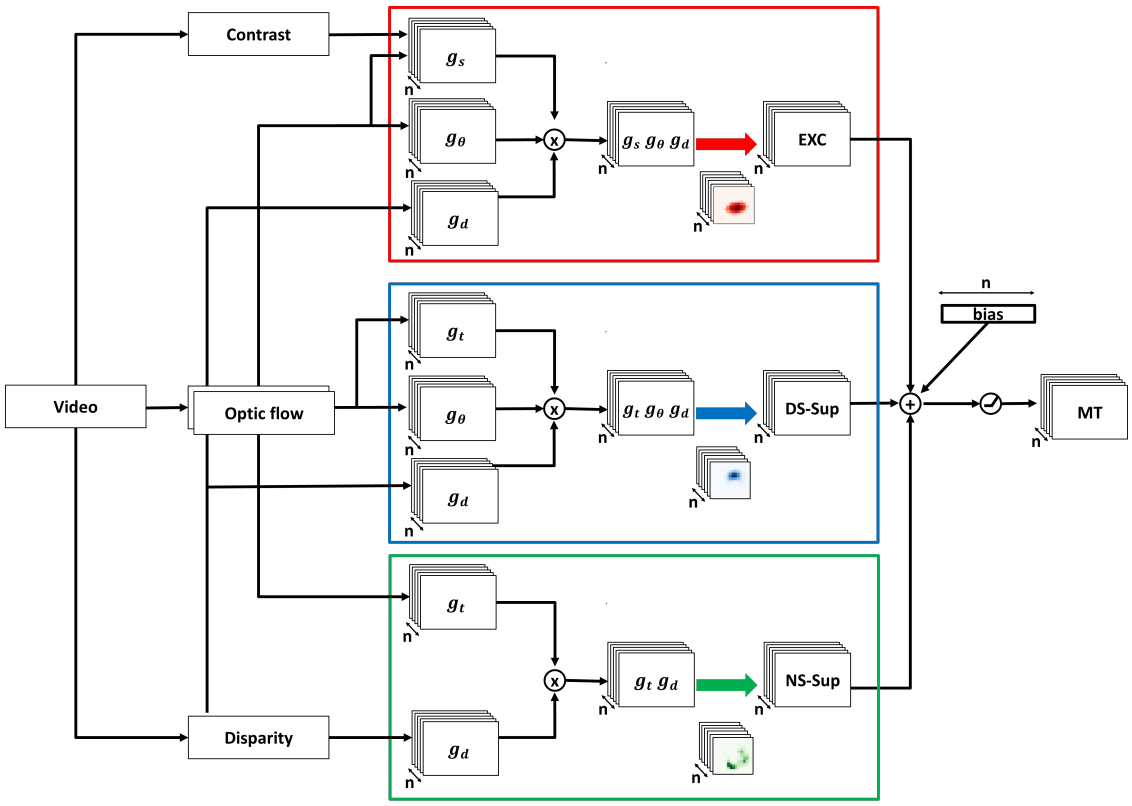


Figure 6.2: Structure of MT Model with three components: Excitation (red box), direction-selective surround suppression (blue box), and non-direction-selective surround suppression (green box). g_s represents speed tuning (Equation 4.8) of the excitation component, g_θ (Equation 4.11) represents direction tuning of the excitation and direction-selective suppression, g_d (Equation 4.12) represents disparity tuning of all components, and finally g_t (Equation 6.3) represents speed tuning of both of the suppression components.

on them. The summation was followed by adding a bias (for each feature map) and passing the result through a ReLU activation to get the final response at the feature maps of the MT layer. The responses of the nodes in these feature maps corresponded to MT activity.

My design was flexible in that it allowed the combination of the components, connected to the ith feature map of MT, to be learned via backpropagation and gradient descent. This flexibility was the result of how I created the sparse convolutional kernels, which were applied on the product maps. More specifically, W_E, W_{DS}, and W_{NS} were, respectively, the sparse kernels for excitation, direction-selective suppression, and non-direction selective suppression.

To create these sparse kernels, I pursued one of the following approaches. In the first approach, I created three "channel-selector" matrices G_E, G_{DS}, and G_{NS} where each column of a matrix was a Gaussian function of n length. The Gaussian functions were all narrow ($\sigma = 0.3$), with trainable mean locations. Multiplying these channel-selector matrices with regular non-sparse convolutional kernels W'_E, W'_{DS}, and W'_{NS} created the sparse kernels that I wanted. More formally,

$$\begin{aligned} W_E &= W'_E \circ G_E, \\ W_{NS} &= W'_{DS} \circ G_{NS}, \\ W_{DS} &= W'_{NS} \circ G_{NS}, \end{aligned} \tag{6.1}$$

where W_E, W'_E, W_{DS}, W'_{DS}, W_{NS}, and $W'_{NS} \in R^{k \times k \times n \times n}$ and G_E, G_{DS}, and $G_{NS} \in R^{n \times n}$. $\circ$ denotes broadcast entry-wise product.

In the second approach, I created three $n \times n$ matrices of trainable parameters. I then applied the softmax function on each column of these matrices. That resulted in three matrices S_E, S_{DS}, and S_{NS}. Similar to Equation 6.1, the sparse kernels were then calculated as,

$$W_E = W'_E \circ S_E,$$
$$W_{NS} = W'_{DS} \circ S_{NS}, \tag{6.2}$$
$$W_{DS} = W'_{NS} \circ S_{NS}.$$

The first approach added $3 \times n$ trainable parameters to the network while the second one added $3 \times n^2$ parameters. However, the first approach was more restricted because the final "selected channel" would mostly be limited to one of the channels near the initialized mean location of the Gaussian. Testing both approaches, I found that the network performance did not suffer due to this limitation mainly because there was no specific order in the parameters of the channels within the product maps and the mean locations were initialized uniformly in $0 - n$ range.

6.2.3 Tuning Fields

The tuning fields were pixel-wise nonlinear functions of optic flow, disparity, and contrast fields. The exact same nonlinear functions, which I discussed in Section 4.2.3, were used to calculate the direction, disparity, and speed tuning of the excitation component (red box in Figure 6.2). The same nonlinear functions were used for the direction and disparity tuning of the surround components (blue and green boxes in Figure 6.2)[2]. However to be consistent with Cui et al. [49], I used a linear combination of ten basis functions as the speed tuning functions of the surround components,

$$g_t(s) = \Sigma_{n=1}^{10} a_n \Lambda_n(s), \tag{6.3}$$

where s is motion speed, Λ_ns were chosen to be overlapping triangle functions with equally spaced centres on a logarithmic scale, and a_ns were trainable parameters (weights) determined during training.

[2]Note that Cui et al. [49] used the same von-Mises function for direction tuning and did not have disparity tuning in their model.

Layer	# Kernels	Kernel Size	Shape	Pool	Nonlinearity
MT	64	15×15	76×76	None	ReLU
Conv-1	64	15×15	76×76	6×6	ReLU
Conv-2	64	9×9	12×12	None	ReLU
Conv-3	64	9×9	12×12	3×3	ReLU
Dense			1024		ReLU
Output			3		None

Table 6.1: Structure of the CNN that was used in the visual odometry task.

Parameter Distributions

Calculation of the tuning fields required tuning parameters. I drew these non-trainable parameters from modelled distributions as explained in Section 4.2.4.

6.2.4 Architecture of the Networks

The odometry network consisted of an MT module (Figure 6.2) followed by three convolutional layers and two dense layers. Table 6.1 summarizes the odometry network parameters. I used 15×15 kernel size for MT module and the first convolutional layer after MT. I chose this size so that after training, kernels in MT module can directly correspond to those of Cui et al. [49] as they used the same size for their kernels. The other network hyperparameters including other kernels sizes and number of channels in each layer were found experimentally to give the best possible performance.

The gesture recognition network also contained an MT module (Figure 6.2) followed by three convolutional layers, a long short-term memory (LSTM) layer, and a dense layer.

Layer	# Kernels	Kernel Size	Shape	Pool	Nonlinearity
MT	64	15×15	$12 \times 76 \times 76$	None	ReLU
Conv-1	64	15×15	$12 \times 76 \times 76$	6×6	ReLU
Conv-2	64	9×9	$12 \times 12 \times 12$	None	ReLU
Conv-3	64	9×9	$12 \times 12 \times 12$	3×3	ReLU
LSTM			256		ReLU
Output			27		Softmax

Table 6.2: Structure of the CNN that was used in the gesture recognition task.

Table 6.2 summarizes the gesture recognition network parameters. The same kernel sizes as the odometry network were chosen. Note that the shape of all layers preceding LSTM layer is three-dimensional where the first dimension is the temporal dimension (see Section 6.2.1). All the convolutional kernels however are two dimensional i.e., the kernels are shared across the temporal dimension.

6.2.5 Training

I used the mean-square error (MSE) loss for the odometry task and the cross-entropy loss for the gesture recognition task. More specifically, the odometry loss was calculated as,

$$E = \sum_{v} (y_v - t_v)^2, \tag{6.4}$$

where t_vs are target velocities and y_vs are network outputs. The gesture recognition

loss was calculated as,

$$E = -\sum_{c} t_c \, log(y_c), \tag{6.5}$$

where t_c is the true probability value for class c and y_c is the predicted probability for that class.

I chose the Adam algorithm [107] as the optimizer mostly with the default parameters (see Section 6.2.7 for exceptions). I used Batch Normalization [99] in the convolutional layers and 50% recurrent Dropout [199] in the LSTM layer. I implemented the networks in Keras [44] using TensorFlow [1] as a backend, and trained them on two NVIDIA GeForce GTX 1080 Ti GPUs.

6.2.6 Spatial Profiles of Suppression

To compare the similarity of the surround kernels of a channel, I used Pearson correlation,

$$\text{Correlation} = \rho(\mathbf{V}_{DS}, \mathbf{V}_{NS}), \tag{6.6}$$

where $\mathbf{V}_{DS}$ and $\mathbf{V}_{NS}$ are the direction-selective and non-direction-selective surround kernels, reshaped to one-dimensional vectors.

I also calculated the same index that was used by Cui et al. [49] to reflect the spatial dispersion of suppression. This index was calculated as,

$$\text{Dispersion} = -\frac{\sum_{x,y} \mathbf{w}(x,y) \sqrt{(x-x_c)^2 + (y-y_c)^2}}{\mathrm{N}\|\mathbf{w}\|}, \tag{6.7}$$

where $\mathbf{w}(x, y)$ was the value of a suppression kernel at position (x, y), (x_c, y_c) was the centre of mass of the kernel, N was the number of values within a kernels (e.g., $15 \times 15 = 225$), and $\|\mathbf{w}\|$ was the norm of the kernel.

6.2.7 Replacing Task-Optimized Surrounds with MT-Like Surrounds

Training each deep network on its respective task resulted in three sets of kernels (excitation (Exc), direction-selective suppression(DS-Sup) and non-direction-selective suppression (NS-Sup)) for MT, which I refer to as task-optimized kernels. These task-optimized kernels had somewhat different statistics (see Results) compared to those found by Cui et al. [49], where the optimization goal was to match MT cell responses. I investigated how replacing the task-optimized surround kernels with the MT-like surround kernels affected performance.

To this end, I augmented the MT-like surround kernels from ten (reported by Cui et al. [49]) to 26460. More specifically, I shifted the base (ten) kernels both horizontally and vertically (shift values were $[-3\colon 1\colon 3]$ pixels), rotated them (rotation values were $[-180\colon 20\colon 180]$ degs), and scaled them (scale values were $2^{[-0.5,0,0.5]}$).

For each task-optimized kernel, I then found the most similar surround kernels from this augmented set. I used correlation as the similarity metric. More specifically, given a pair (i.e., DS-Sup and NS-Sup) of task-optimized surround kernels I did 26460 comparisons between that pair and each pair of the augmented MT-like set and chose the pair (from the augmented set) with the highest pair-average correlation with the task-optimized pair. Finally, each kernel of the chosen surround was rescaled to have the same mean as its corresponding task-optimized surround kernel.

After replacing task-optimized kernels with MT-like kernels, I fine-tuned the networks for five epochs. I used 2×10^{-5} as the learning rate of the surround kernels and 10^{-4} for the rest of the network.

6.2.8 Motion-Opponency Model

Cui et al. [49] found that adding both types of surrounds to their framework resulted in more accurate prediction of neural data compared to a one-component framework with no

surround. Such a one-component model is referred to as a motion-oponency model because it can only predict suppression when stimuli move in the null-direction of a neuron (see Section 4.2.3) but not when stimuli appear in the surround of the classical receptive field.

To test the functional role of the surrounds and how much they contribute in performance, I also created surround-less networks where I removed both surround components from the MT framework and trained the networks from scratch. I call these networks, motion-oponency networks.

6.2.9 Surround-Suppression Strength of Following Convolutional Layers

I investigated the surround-suppression strength of the convolutional layers after the MT part of the network. For this investigation, I first created random dots moving with different speeds ([1, 2, 4, 8, 16, 32, 64] deg/s), different directions ([0 : 15 : 345] deg), and different sizes ([8, 16, 32, 64, 76] pixels). For each neuron, I found the maximum of the three-dimensional tuning curve (speed, direction, and size) and created a one-dimensional size tuning curve passing from the maximum. Then using the size tuning curve, the strength of the surround was calculated as,

$$\frac{t_{max} - t_{76}}{t_{max}}, \tag{6.8}$$

where t_{max} is the maximum value of the size-tuning curve and t_{76} is the value of the curve at size=76, which was the size of the input frames.

To investigate the effect of MT surrounds on surround-suppressive strength of the following convolutional layers, I measured their strength before and after removing MT surrounds from the excitation-suppression networks with task-optimized surrounds and from the excitation-suppression networks with MT-like surrounds. Note that these networks were trained having MT surrounds and no further training was conducted after surround elimination.

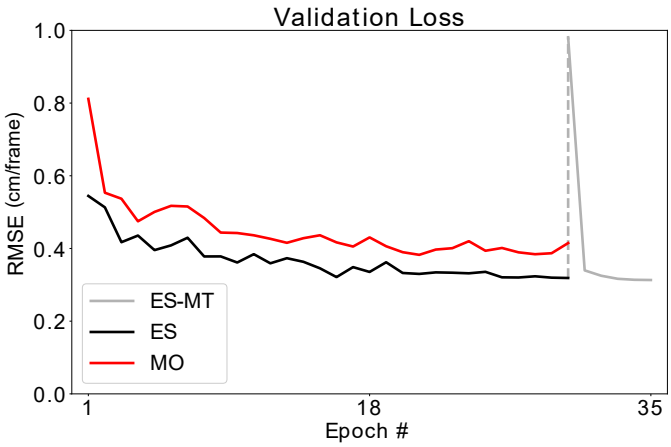


Figure 6.3: Validation loss curves for the odometry task. ES: excitation-suppression network with task-optimized surrounds. ES-MT: excitation-suppression network with MT-like surrounds. MO: motion-opponency (excitation only) network. The standard deviation of the target velocities of the validation set was 1.54 cm/frame.

6.3 Results

Figure 6.3 shows the validation-loss curves of the odometry task for three different networks with identical architecture (except the MT module; see Table 6.1): the excitation-suppression network with task-optimized surrounds (ES), the excitation-suppression network with MT-like surrounds (ES-MT), and the excitation-only or motion-opponency network (MO). Both ES and ES-MT had three components (one excitation and two suppressive surrounds) in their MT module, while MO had only the excitation component. Both ES and MO were trained for thirty epochs. ES-MT, which was the result of replacing the surround kernels of the MT module in ES with MT-like surround kernels (see Section 6.2.7), was fine-tuned for five epochs. Figure 6.3 shows that after replacing the kernels in ES-MT, the error went up but after a few epochs came down to the same level as ES. The lowest mean-square error of MO was about 20 percent higher compared to the other networks.

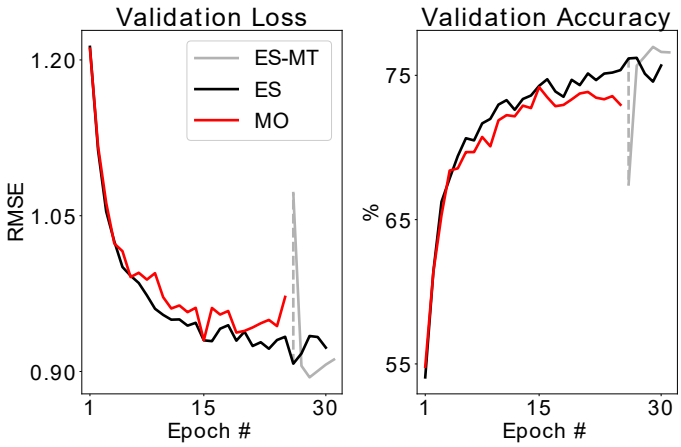


Figure 6.4: Left, validation loss curves for gesture recognition task. Right, validation accuracy curves for gesture recognition task. ES: 3-component MT model. ES-MT: 3-component MT model with replaced kernels from MT data. MO: excitation only MT model.

Figure 6.4 illustrates the validation-loss and validation-accuracy curves for the gesture recognition task for the ES, ES-MT, and MO networks where all had identical architecture (see Table 6.2), except MO had only the excitation component in its MT module. Both ES and MO were trained for thirty epochs. ES-MT was fine-tuned for five epochs after replacing the surround kernels of the MT module. ES-MT achieved about 1 percent better accuracy, compared to ES, after fine-tuning. On the other hand, the highest achieved accuracy of MO was about 2 percent worse than ES.

Figure 6.5 illustrates six examples of MT kernels in the ES and ES-MT odometry networks. Top rows are the ES kernels that were found by training the network for odometry. Bottom rows are the ES-MT kernels. These were the result of replacing the ES surround kernels with MT-like surrounds (see Section 6.2.7), and fine-tuning the network. Note that fine-tuning did not noticeably change MT-like kernels due to the quite small learning rate, which were used. In other words, the structure of the MT-like kernels before and after fine-tuning was highly similar ($r > .95$).

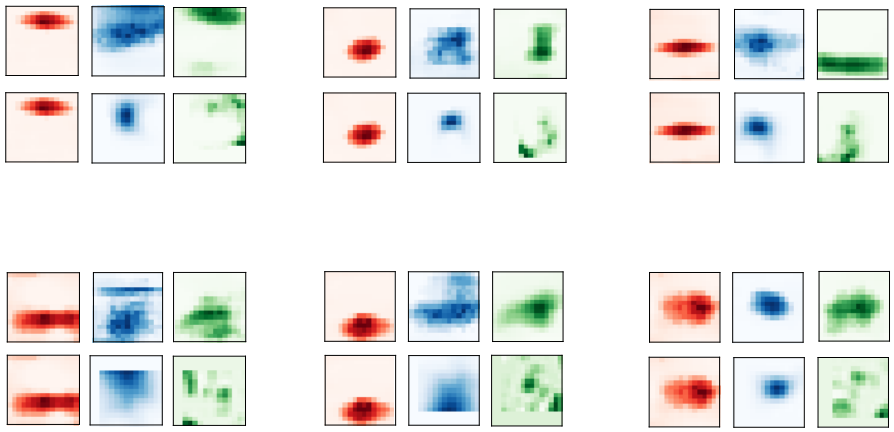


Figure 6.5: Six examples of MT kernels of the odometry networks. Each 2×3 group shows one channel where top row illustrates task-optimized kernels and bottom row illustrates MT-like kernels. Red is excitation kernel (which was not replaced), blue is direction-selective suppression surround kernel (DS-Sup) and green is non-direction-selective suppression (NS-Sup) surround kernel. The correlation between all of the sixty four best-matched surround kernels and the task-optimized surround kernels was 0.48 ± 0.08 (mean$\pm$SD), while a random match gave 0.05 ± 0.12. Generally, a better match was found for the DS-Sup kernels where the average correlation was 0.68 vs NS-Sup average of 0.12.

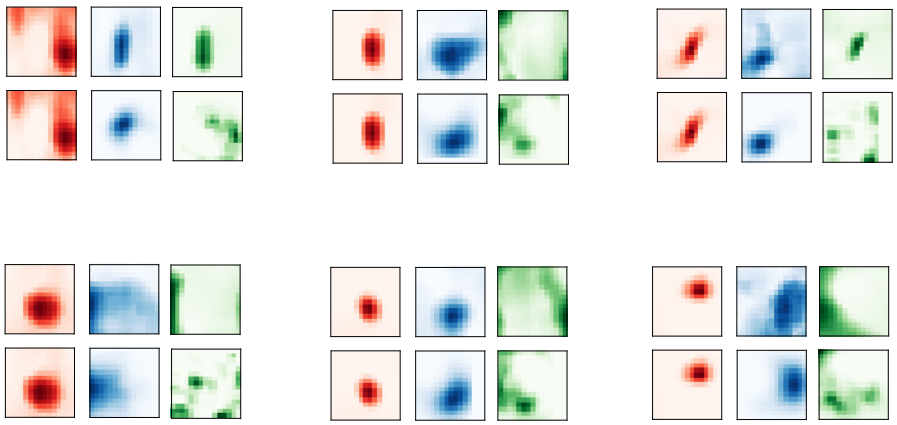


Figure 6.6: Six examples of MT kernels of gesture recognition networks. Each 2×3 group shows one channel where top row illustrates task-optimized kernels and bottom row illustrates MT-like kernels. Red is excitation kernel (which was not replaced), blue is direction-selective suppression surround kernel (DS-Sup) and green is non-direction-selective suppression (NS-Sup) surround kernel. The correlation between all of the sixty four best-matched surround kernels and the task-optimized surround kernels was 0.51 ± 0.09 (mean$\pm$SD), while a random match gave 0.06 ± 0.15. Generally, a better match was found for the DS-Sup kernels where the average correlation was 0.72 vs NS-Sup average of 0.14.

Figure 6.6 is similar to Figure 6.5 except it illustrates the MT kernels of ES and ES-MT networks for the gesture recognition task.

Figures 6.7 and 6.8 show a comparison between the spatial statistics of the task-optimized and MT-like surround kernels in odometry and gesture recognition networks, respectively. The left panels are histograms of correlations between the direction-selective-suppressive (DS-Sup) and non-direction-selective-suppressive (NS-Sup) surround kernels. While some task-optimized surround kernels had high correlations indicating similar shapes (for example see the top left panel in Figure 6.6), none of the MT-like surround pairs had

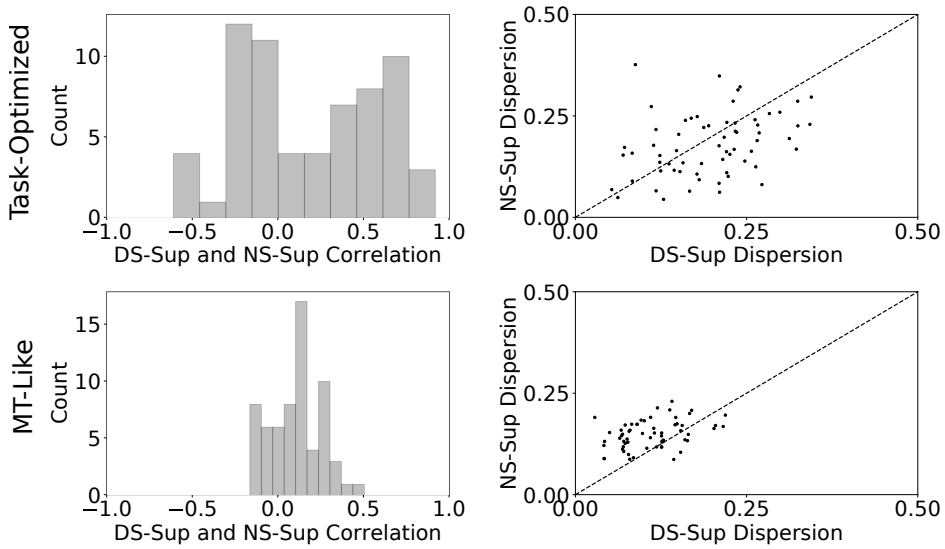


Figure 6.7: Comparison between task-optimized and MT-like surround kernels in the odometry networks. Left, histogram of correlation between direction-selective-suppressive (DS-Sup) and non-direction-selective-suppressive (NS-Sup) surround kernels for task-optimized network (top) and MT-like (bottom) network. Right, scatterplots of dispersion of DS-Sup vs NS-Sup surround kernels with task-optimized at the top and MT-like at the bottom. The MT-like dispersion scatterplot shows that MT-like surround kernels had similar statistical relationship as those found by Cui et al. [49] (see their Figure 5-E.)

similar shapes. The statistical difference is also evident from the scatterplots (right panels of Figures 6.7-6.8) where in the case of MT-like kernels (bottom-right panels) the dispersion of an NS-Sup surround was almost always higher than the dispersion of its corresponding DS-Sup surround.

Figures 6.9, 6.10, and 6.11, respectively, illustrate the box-whisker plots of surround-suppression strength of odometry (left panels) and gesture recognition networks (right panels) in the first, second, and third convolutional layers, following the MT module.

For the first convolutional layer, the average surround-suppression strength of MO net-

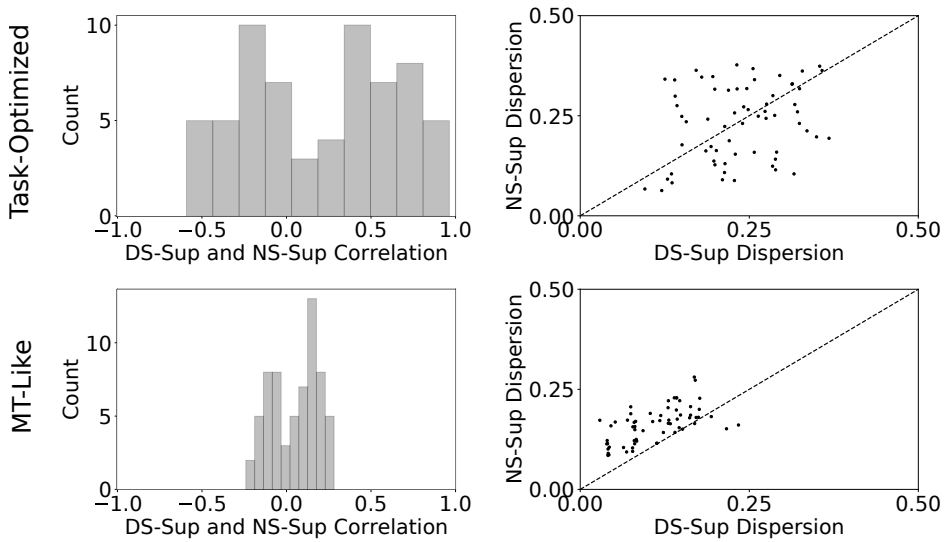


Figure 6.8: Comparison between task-optimized and MT-like surround kernels in the gesture recognition networks. Left, histogram of correlation between direction-selective-suppressive (DS-Sup) and non-direction-selective-suppressive (NS-Sup) surround kernels for task-optimized network (top) and MT-like (bottom) network. Right, scatterplots of dispersion of DS-Sup vs NS-Sup surround kernels with task-optimized at the top and MT-like at the bottom. The MT-like dispersion scatterplot shows that MT-like surround kernels had similar statistical relationship as those found by Cui et al. [49] (see their Figure 5-E.)

works is significantly higher than the average surround-suppression strength of ES and ES-MT networks ($p < 0.1$, t-test). Therefore, it seems that MO networks, which were trained without MT surrounds, learned to compensate for the lack of surrounds by introducing stronger suppressive surrounds in later layers.

Generally, removing the surround kernels from the MT module of ES and ES-MT networks (labelled as ES-Cut and ES-MT-Cut in the figures) caused the surround-suppression strengths to decrease in the following convolutional layers[3]. This can be well seen in Figure 6.10 and particularly in Figure 6.11 where the average strengths of ES-Cut compared to ES and ES-MT-Cut compared to ES-MT were significantly lower ($p < 0.1$, t-test). Note that MO, ES, and ES-MT networks had strong surround suppression at this level.

6.4 Discussion

In this chapter, I investigated whether the fine structure of MT surrounds is related to task performance. For this investigation, I first directly optimized the surrounds for task performance. I found that these task-optimized surrounds were statistically different from MT surrounds reported by Cui et al. [49]. More specifically, the dispersion relationship between task-optimized DS-Sup and NS-Sup pairs seemed arbitrary, as opposed to higher dispersion for NS-Sup against DS-Sup in MT surrounds. On the other hand, some task-optimized surround pairs looked similar.

Next, I replaced the task-optimized surrounds with more MT-like surrounds. After fine-tuning, networks with the MT-like surrounds had similar performance as those with task-optimized network. This suggests that a fairly large family of surround structures is essentially equally consistent with both the odometry and gesture recognition tasks. In other words, while the gradient descent and the brain find structurally different surround kernels for the tasks, both are similarly effective solutions. However, it is important to note

[3]Note that after removing MT surrounds, ES-Cut and ES-MT-Cut networks were not trained any further.

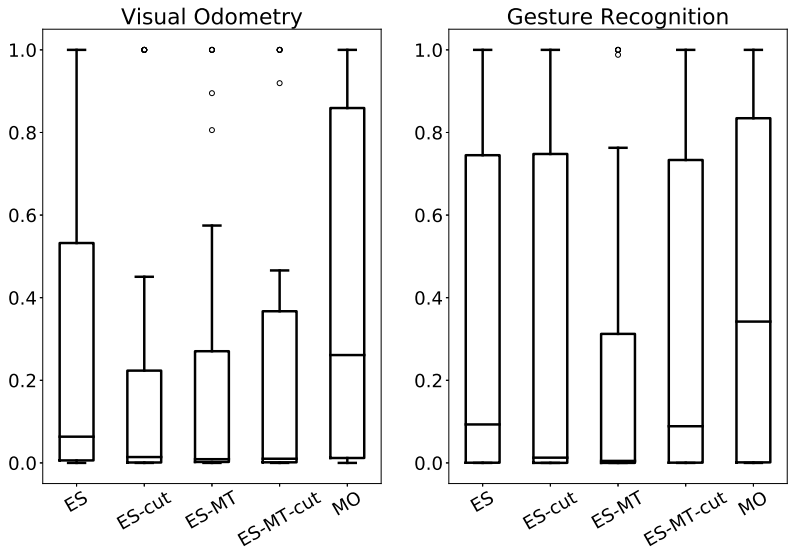


Figure 6.9: Box-whisker plots of surround-suppression strength in the first convolutional layer after MT module. ES: excitation-suppression networks with task-optimized surrounds. ES-Cut: excitation-suppression networks with task-optimized surrounds after removing surround components from MT module. ES-MT: excitation-suppression networks with MT-like surrounds. ES-MT-Cut: excitation-suppression networks with MT-like surrounds after removing the surround components from MT module. MO: motion-opponency (excitation only) networks.

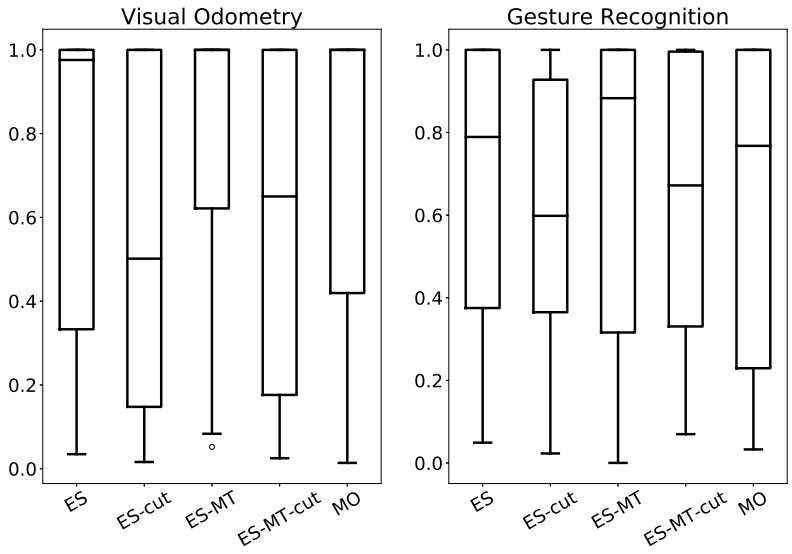


Figure 6.10: Box-whisker plots of surround-suppression strength in the second convolutional layer after MT module. ES: excitation-suppression networks with task-optimized surrounds. ES-Cut: excitation-suppression networks with task-optimized surrounds after removing surround components from MT module. ES-MT: excitation-suppression networks with MT-like surrounds. ES-MT-Cut: excitation-suppression networks with MT-like surrounds after removing the surround components from MT module. MO: motion-opponency (excitation only) networks.

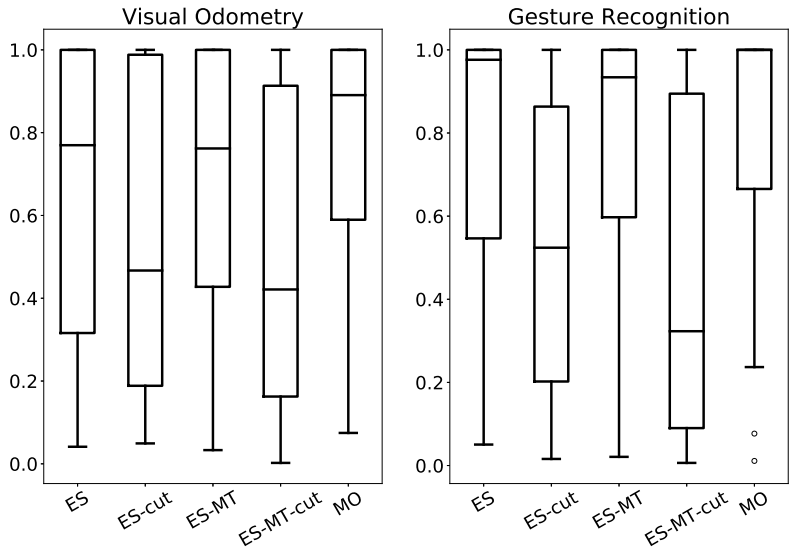


Figure 6.11: Box-whisker plots of surround-suppression strength in the third convolutional layer after MT module. ES: excitation-suppression networks with task-optimized surrounds. ES-Cut: excitation-suppression networks with task-optimized surrounds after removing surround components from MT module. ES-MT: excitation-suppression networks with MT-like surrounds. ES-MT-Cut: excitation-suppression networks with MT-like surrounds after removing the surround components from MT module. MO: motion-opponency (excitation only) networks.

that MT is involved in a range of tasks. Therefore, the gradient descent solution might become more similar to that of the brain if a network is trained to simultaneously solve a range of MT-related tasks as opposed to only one. Having a network architecture that resembles the dorsal stream more closely may also result in a more similar solution.

The networks that did not include any surround in their MT module (MO networks) performed nearly as well as the networks with MT surrounds. In such networks, the lack of surround in MT seemed to be compensated by learning to have stronger surround suppressions in higher layers. This functional adaptation is consistent with lesion studies in MT where deficits in motion perception or smooth-pursuit eye movement, caused by small or moderate-sized lesions, have been reported to be transitory, with almost complete recovery within days [137].

Finally, removing MT surrounds (without further training) from the ES or ES-MT networks, which were trained with MT surrounds, impaired surround suppression in the higher-lever convolutional layers of these networks. This indicated that higher-level layers inherited their surround suppression from a lower (e.g., MT) layer.

6.4.1 Future Work

While the source of different forms of suppression remains unclear in the brain [49], using the presented framework may help to gain some clarity. For example, consider a case where local contrast normalization, which corresponds to lateral inhibitions, is added to the ND-Sup component of a network. Let us assume that ND-Sup kernels, after training, acquire more similar spatial properties to those of MT-like kernels. Such hypothetical scenario may hint that lateral inhibitions are the source behind NS suppression.

Chapter 7

Guiding Deep Representations with an Empirical Model of MT

7.1 Introduction

Another potential application of the empirical MT model is discussed in this chapter i.e., to make the internal representations of deep networks more physiologically realistic. In general, deep learning may facilitate development of visual cortex models with more ethologically realistic functions. For example, various deep networks excel in scene segmentation [40, 41, 83], depth estimation from stereo disparity [228], and scene flow [134]. The internal visual representations of deep networks that have been trained for object recognition have striking relationships with representations in the ventral stream [225, 36, 105, 92, 226] (relatedly, Güçlü and van Gerven [81] found that action-recognition CNNs were predictive of function magnetic resonance imaging data from the dorsal stream), but there are also striking differences [214].

The empirical MT model may provide regression targets for intermediate network layers, helping to impose a physiologically realistic representation. A related approach was taken by Arai et al. [10], who optimized a two-layer network model of superior colliculus with two

cost terms, one (applied to the output) related to the task, and the other (applied to the hidden layer) derived from neural activity. More recently, Yamins et al. [225] trained deep networks to approximate recordings of neurons in the inferotemporal (IT) cortex. The resulting networks accounted for much of the variance in held-out inferotemporal neural data. Interestingly, IT predictions of similar quality were obtained simply by training the networks for object recognition, although the neural dataset was small enough (5760 images) that overfitting was possible in this case. Other groups have reported good results training deep networks to emulate neural recordings in the retina [135], V1 [106, 108, 35], and V4 [151]. McIntosh et al. [135] found that a deep network generalized better across stimulus types than other models, and also reproduced sub-Poisson noise scaling found in the retina. Tripp [213] previously trained an intermediate layer of a deep network, which had motion-energy components in a lower layer, to emulate a simplified empirical model of MT activity, and then trained the full network to estimate self-motion speed and direction from video. I extend this approach with the present (more detailed) empirical model (see Chapter 4). This approach has advantages and disadvantages compared to using MT data directly for regression targets. The main disadvantage is reduced physiological validity. Advantages are the possibility of unlimited training data, and the ability to directly manipulate tuning statistics, to allow detailed exploration of the relationships between representations and behaviour.

7.2 Methods

Visual odometry is the process of using visual information to estimate self-motion, a function that involves the dorsal stream. Neurons in the middle superior temporal area (MST), which receives strong input from MT, respond to large optic flow patterns such expansion and spiral motion, which are highly relevant to self-motion (see Section 2.1.3).

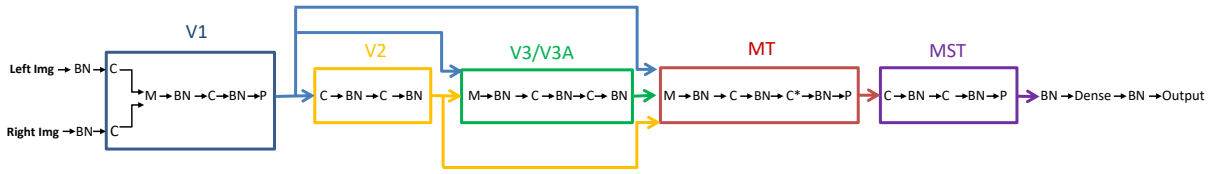


Figure 7.1: Structure of the CNN. BN: Batch Normalization, M:merge layer, C: Convolutional layer, P: pool layer. *The MT cost is applied at the output of the second convolutional layer in MT.

7.2.1 A Novel Visual Odometry Dataset

To train the deep CNN, I used the novel visual odometry dataset, which I had synthetically created in Unreal Engine 4 (see Section 5.2.1). More specifically, the deep CNN took a stereo sequence as input (for each move), and the corresponding medio-lateral, antero-posterior, and angular velocities as regression targets for the output layer. To train the middle layer of the deep CNN, corresponding to area MT, I calculated dense direction, speed, and disparity fields (pyramidal Lucas-Kanade method) as well as contrast fields (Peli method) for every frame of the sequence. For each of these four fields, I fed the sequence-average field to the empirical model, to produce regression targets for the MT layer. The target therefore reflects average stimulus features over several frames, roughly consistent with the low-pass properties of MT neurons [12].

7.2.2 Architecture

I created a deep convolutional network that was based loosely on the macaque dorsal visual stream. The network architecture is shown in Figure 7.1, and Table 7.1 lists the

Layer	# Kernels	Kernel Size	Shape	Pool	Nonlinearity
V1-1	256	7×7	70×70	None	ReLU
V1-2	256	7×7	70×70	None	ReLU
V1-binocular	256	7×7	64×64	3×3	ReLU
V2-1	256	7×7	21×21	None	ReLU
V2-2	256	7×7	21×21	None	ReLU
V3-1	128	7×7	21×21	None	ReLU
V3-2	128	7×7	21×21	None	ReLU
MT-1	64	5×5	17×17	None	ReLU
MT-2	64	5×5	13×13	2×2	ReLU
MST-1	128	9×9	6×6	None	ReLU
MST-2	128	9×9	6×6	2×2	ReLU
Dense			1024		ReLU
Output			3		None

Table 7.1: Structure of the example CNN that I used in the visual odometry task. The kernel sizes had been experimentally selected to give the best odometry performance. It is important to note that in a CNN the receptive field (RF) size of the network layers increases as we move towards the output (regardless of each layer's kernel size). So for example, although MT layers had 5×5 kernel size, they had larger RF sizes compared to earlier V1 layers with 7×7 kernel size.

network parameters. The left and right input layers (stereo frames) were each followed by a convolutional layer. I then merged these two layers together and connected the result to the third convolutional layer. This convolutional layer was followed by a max-pooling layer. These layers correspond roughly to the primary visual cortex (V1), which includes binocular neurons and complex cells. I used two convolutional layers to model each of V2, V3/V3a, MT, and MST, to model a separation between input and output cortical layers in each area. I added skip-connections consistent with [126]. Specifically, the first convolutional layer corresponding to V3/V3a received input from the last layers of both areas V1 and V2, and the first convolutional layer of area MT received input from the last layer of all earlier areas. After the MST layers, I added a dense layer with 1024 hidden units and an output layer with three units to estimate medio-lateral, antero-posterior and angular velocities from input frames[1].

7.2.3 Training

To train the deep network to both approximate odometry and emulate the empirical model I pursued two approaches. In the first approach, I first trained the network up to MT-2 layer (see Table 7.1) to only approximate MT activity, for forty epochs. I used the root-mean-square error of MT-2 layer outputs and the MT activity targets as the training loss. These target activities were calculated with a simplified version of the empirical model where the dynamics of pattern and component selectivity and motion-in-depth tuning were omitted, and I chose difference of Gaussian kernels as receptive fields. Each of these kernels was elongated orthogonal to the preferred direction of its respective unit. After training for MT activity, I "froze" these layers and trained the rest of the network (i.e., from MST-1 layer to the end) for odometry task, for forty epochs. Here I used the root-mean-square error

[1]Note that this network architecture does not account for two of the visual processing centres in the primate brain prior to V1, the retina and the lateral geniculate nucleus of the thalamus. Not accounting for these processing centres might contribute to inability of producing a deep CNN with MT-like representation and good performance (see Section 7.4).

of the network outputs and velocity labels in the novel odometry dataset as the training loss. Finally to achieve a better performance on the odometry task, I unfroze all network layers and trained it only on odometry for another five epochs.

In the second approach, I trained the network on both odometry and MT activity simultaneously. In this case, the training loss was a linear combination of both MT activity and odometry losses. This combined loss function can be written as,

$$E = A_1 \sum_v (y_v - t_v)^2 + A_2 \sum_i (y_i - r_i)^2, \tag{7.1}$$

where t_vs are target velocities, y_vs are network outputs, r_is are normalized neural responses, computed using the empirical model on input frames, and y_is are unit activities of MT-2 feature maps. Finally, A_1 and A_2 are linear weights.

I implemented the convolutional network in Keras [44] using TensorFlow [1] as a back-end, and trained it on an NVIDIA GeForce GTX Titan Xp GPU.

I used the Adam algorithm [107] as the optimizer with the default parameters. I also used Batch Normalization [99] in some layers (see Figure 7.1). Like Dropout [199], Batch Normalization has regularization benefits, which reduces overfitting, while it also speeds up training.

7.3 Results

7.3.1 Odometry Performance

I trained convolutional neural networks (CNNs) to estimate self-motion from visual input, as described in the Methods. The dataset included naturalistic visual stimuli, but since the dataset was synthetic, I had ground-truth velocity labels. I used the empirical model for MT labels, but I omitted the dynamics of pattern and component selectivity, as emulating these dynamics might require a more complex recurrent network. Figure 7.2 shows the

validation loss curves of two different networks. CNN-O network was trained only on the odometry task (no emulation of MT responses). CNN-OMT was trained with a linear combination of both costs (Equation 7.1). I chose $A_1 = 1$ and tested different values for A_2. I found $A_2 = 4$ to be the best choice, as larger values prevented the combined validation loss from going down, and smaller values made the second cost negligible compared to the first. I also trained a third network, CNN-3Phases, in three phases (as described in Section 7.2.3): I first trained the part of the network up to MT, with the MT cost (CNN-MT); then the rest of the network with the odometry cost (with the weights up to the MT layer frozen); and finally the full network with the odometry cost. Figure 7.3 shows a scatter plot of actual velocities (of the validation set) against the velocities predicted by these three networks. As the correlations between the network-output and target velocities suggest, all three networks perform quite well.

In Figure 7.4, I show the MT validation loss curves for CNN-OMT and CNN-MT (i.e., the first phase of CNN-3Phases). The loss is higher for CNN-OMT since the network had to learn not only to emulate MT activity targets but also to estimate velocity targets. Although I tried both larger and smaller values for A_2, I could not reduce MT loss any further for CNN-OMT (data not shown). To confirm that this was in fact due to the odometry cost, rather than details of the training approach, I continued training of CNN-OMT with the odometry cost removed. The MT cost then declined rapidly (dashed line).

7.3.2 Speed and Direction Tuning of CNN Units

In this section, I examine speed and direction tuning of units in the MT layers of the three CNNs: one trained for the odometry task alone (CNN-O), one trained for MT response approximation alone (CNN-MT), and one trained with both these cost terms simultaneously (CNN-OMT).

Figures 7.5 and 7.6 show tuning curves of CNN-O (trained for the odometry task alone). Previous work [e.g. 225] has shown that task-trained CNNs often have physiologically relevant representations. Indeed, the CNN-O units have tuning for both direction and

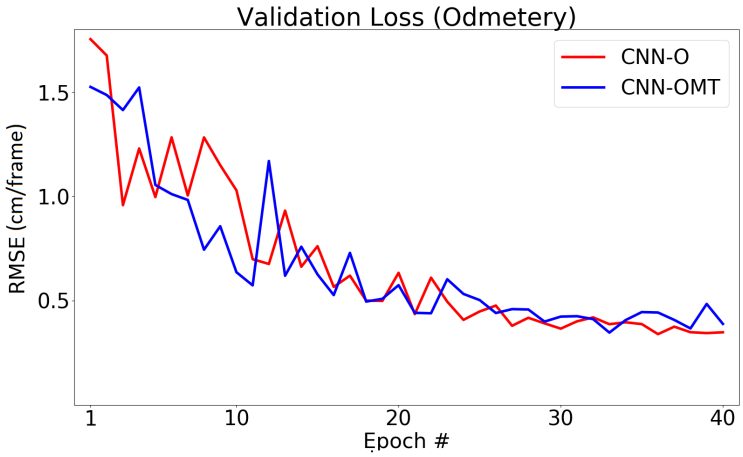


Figure 7.2: Validation loss curves for odometry task in two networks. CNN-O: the network was trained only with odometry cost, CNN-OMT: the network was trained simultaneously with MT and odometry costs. The standard deviation of the target velocities of the validation set was 1.54 cm/frame.

speed of visual motion. This is unsurprising, because the task depends entirely on the pattern of direction and speed across the visual field. However, the tuning curves are somewhat different than physiological tuning curves. Many of the direction-tuning curves are narrow (Figure 7.7), and most of the speed tuning curves are monotonic and high-pass. Also, the tuning curves of many units are quite sensitive to the stimulus used to calculate the curves. In particular, they are quite different for dot stimuli vs. scene stimuli. This difference stems from the fact that the dot stimuli have quite different statistics from those of natural scene stimuli, which the networks have been trained on.

Figures 7.8 and 7.9 show example tuning curves for CNN-MT, along with the target curves for each unit, from the empirical model. Despite fairly low regression error on the validation stimuli, some substantial differences are evident in the tuning curves. This is likely because the distribution of stimuli in the odometry task is different than the distribution of stimuli used to make the tuning curves. For example, in the task stimuli, horizontal motion is represented more strongly than motion in other directions, due to

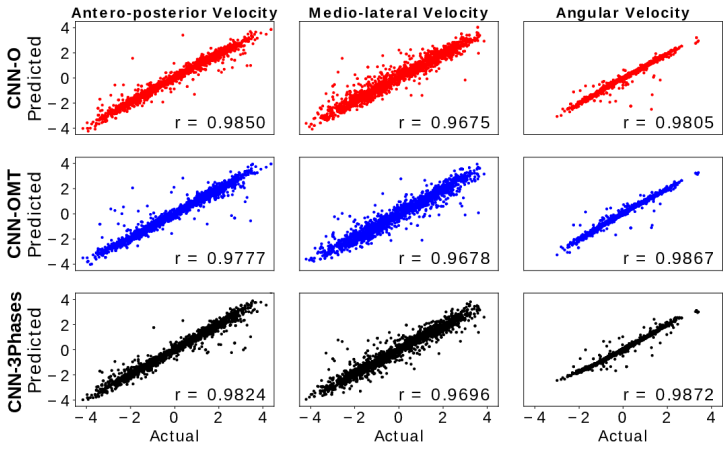


Figure 7.3: Scatter plots of actual vs. predicted self-motion velocities of the validation set. Top: CNN-O, the network only trained with odometry loss. Middle: CNN-OMT, the network trained simultaneously with both MT and odometry losses. Bottom: CNN-3Phase, the network trained in three phases: (1) up to the MT-2 layer with MT loss, (2) after the MT-2 layer with odometry loss, (3) the complete network with odometry loss.

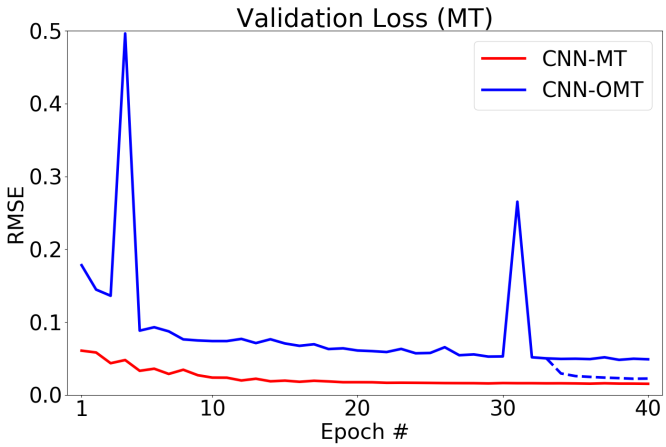


Figure 7.4: Validation loss curves for MT regression. CNN-MT: the network was trained only with MT cost, CNN-OMT: the network was trained with MT and odometry costs. The dashed line shows the training curve of CNN-OMT for seven epochs when I trained only with MT cost (no odometry cost), initialized with 33rd-epoch weights that gave the lowest MT loss.

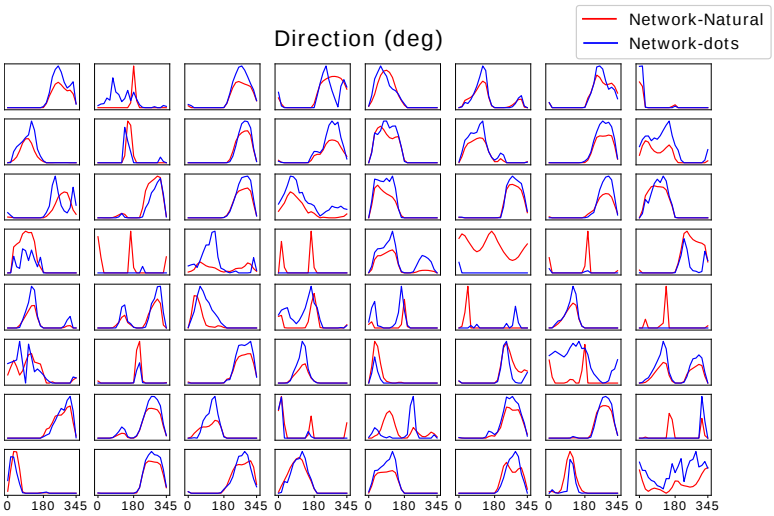


Figure 7.5: Direction-tuning curves of example units in CNN-O. The red and blue curves are responses to natural-scene stimuli and random-dot stimuli, respectively. The networks were trained only on natural-scene stimuli. The direction-tuning curve of each unit is measured at the maximum of 0.5°/s and the speed at which the unit responded most strongly. I used minimum of 0.5°/s because the computer-vision results were less reliable at lower speeds, resulting in noisier tuning curves. The curves are normalized to their peak responses.

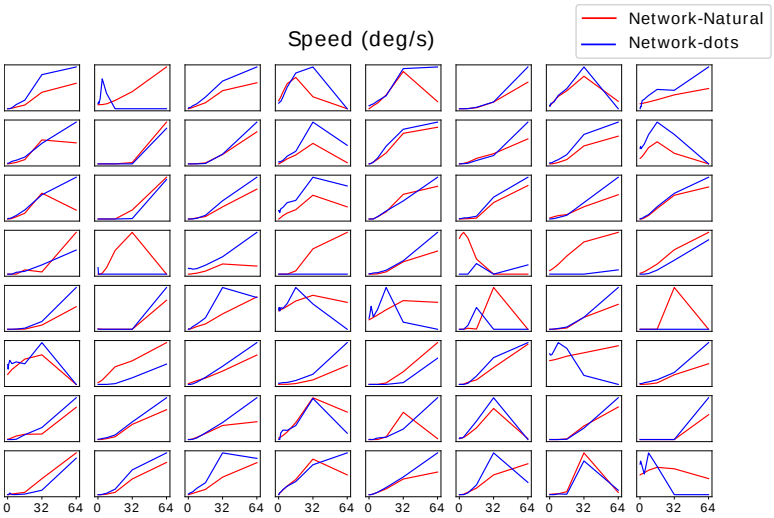


Figure 7.6: Speed-tuning curves of example units in CNN-O (the same units as in Figure 7.5). These were measured at the motion direction that evoked the strongest response. Conventions as in Figure 7.5.

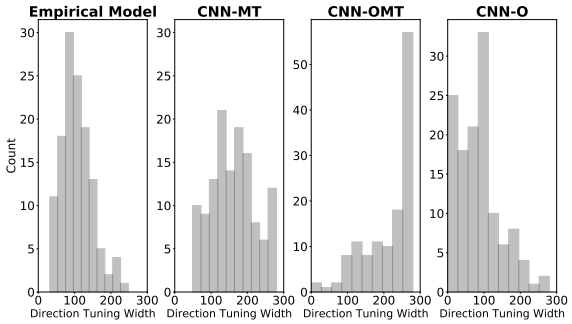


Figure 7.7: Left, half-height widths of direction-tuning curves in the empirical model units that make up the target population for CNN-MT and CNN-OMT. Second left, half-height widths of direction-tuning curves in the MT layer of CNN-MT. Second right, half-height widths of direction-tuning curves in the MT layer of CNN-OMT. Right, half-height widths of direction-tuning curves in the MT layer of CNN-O (trained only for the odometry task). Many of these direction-tuning curves are narrow.

horizontally curvilinear self-motion paths.

The sensitivity of the tuning curves to the stimulus (i.e., random dots vs. natural scenes) is much lower in CNN-MT than CNN-O (Figure 7.12, top vs. middle row), despite the fact that both networks were trained only on natural scenes, and in fact with the same set of stimuli.

Figures 7.10 and 7.11 show example tuning curves of CNN-OMT. This network's tuning was weakly related to the targets from the empirical model. For example, the direction-tuning curves are quite broad (see also Figure 7.7). This is somewhat surprising, because the MT regression error was only moderately higher in this network than in CNN-MT (root mean-squared error .048 vs. 0.015). Low regression cost may be possible, despite poor tuning curves, due to good prediction of low activities, and good prediction for speeds and directions that are most commonly seen in training and testing. This outcome suggests that changes to the regression cost may be needed to produce realistic tuning in this context. Possible changes include training with a different distribution of stimuli, or weighing the cost of rare cases more heavily. This network CNN-OMT was also sensitive to the stimulus (Figure 7.12, bottom row).

Figure 7.13 compares correlations between target and actual tuning curves for CNN-MT (top row), CNN-3Phases (second and third rows), and CNN-OMT (bottom row). These plots show that many tuning curves of CNN-OMT are poorly related to those of the empirical model, particularly for dot stimuli. The same can be said for CNN-3Phases especially as the third training phase (i.e., training only on the odometry task) advanced (first epoch vs. fifth epoch). Also, pursuing different training approaches (see Section 7.2.3) affected the tuning similarities, hence CNN-OMT had higher speed-tuning correlations vs. CNN-3Phases with moderately higher direction-tuning correlations.

One effect on tuning of the additional task cost in CNN-OMT (compared to CNN-MT) was to reduce the depths of the direction and speed tuning curves. When tuning curves were normalized to the peak of their targets, the standard deviation of CNN-OMT direction-tuning curves averaged 0.32 (vs. 0.63 for CNN-MT). Similarly, the standard

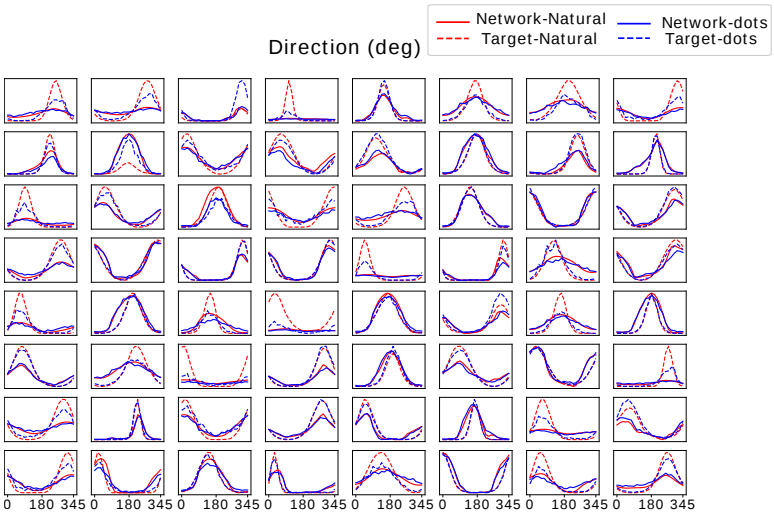


Figure 7.8: Direction-tuning curves of example units in CNN-MT. The red and blue traces are tuning with natural-scene and dot stimuli, respectively. The dashed lines indicate target values from the empirical model. These are slightly different for natural-scene and dot stimuli, due to differences in interpretation by the computer-vision algorithms, and differences in contrast between the stimuli. Similar to Figure 7.5, the direction-tuning curves were measured at the preferred speeds of the empirical model, or 0.5°/s, whichever was greater. Preferred speeds were calculated separately for dot and natural-scene stimuli, based on their mean contrasts. Mean ± SD for correlation between natural-scene stimulus tuning of target (dashed red) and network units (solid red): 0.90 ± 0.08, and for correlation between dot stimulus tuning of target (dashed blue) and network units (solid blue): 0.90 ± 0.07.

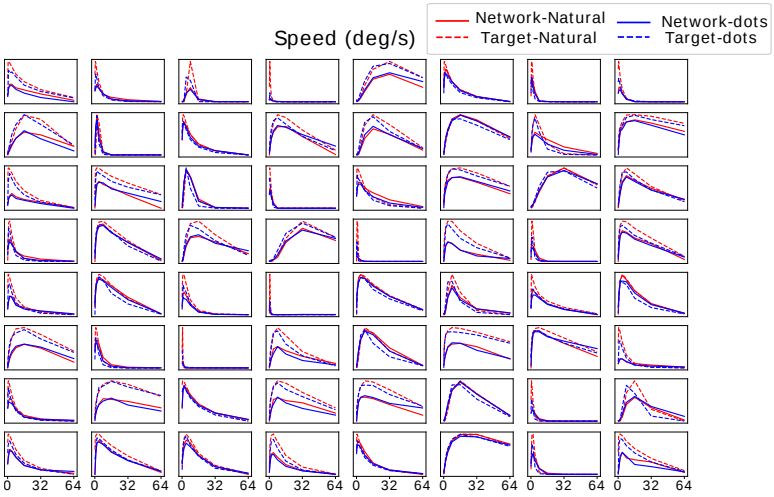


Figure 7.9: Speed-tuning curves of example units in CNN-MT, calculated at the preferred directions of the empirical model units. Conventions as in Figure 7.8. Mean ± SD for correlation between natural-scene stimulus tuning of target (dashed red) and network units (solid red): 0.91 ± 0.10, and for correlation between dot stimulus tuning of target (dashed blue) and network units (solid blue): 0.93 ± 0.05.

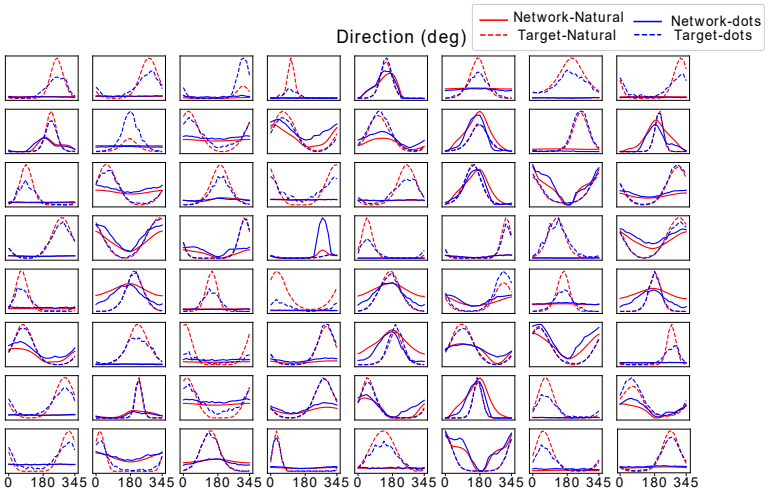


Figure 7.10: Direction-tuning curves of example units in CNN-OMT. Conventions as in Figure 7.8. Mean ± SD for correlation between natural-scene stimulus tuning of target (dashed red) and network units (solid red): 0.57 ± 0.50, and for correlation between dot stimulus tuning of target (dashed blue) and network units (solid blue): 0.55 ± 0.39.

deviation of the normalized CNN-OMT speed-tuning curves averaged 0.71 (vs. 0.67 for CNN-MT). The MT cost affected tuning. For example the speed tuning curves are less uniformly high-pass in CNN-OMT than in CNN-O. However, the MT cost did not make tuning realistic in either CNN-OMT or CNN-3Phases. The two cost terms may have exerted conflicting influences on tuning during training, suggesting either a limitation of the model or the training algorithm, or low specialization of MT for visual odometry.

7.4 Discussion

When I used the empirical model to train convolutional networks, the interaction between the task cost and the MT-regression cost was complex. I trained odometry networks to use the empirical MT model as input (Figures 5.2 and 5.3), and these performed as well as odometry networks with video input (Figure 7.2), indicating that the model of

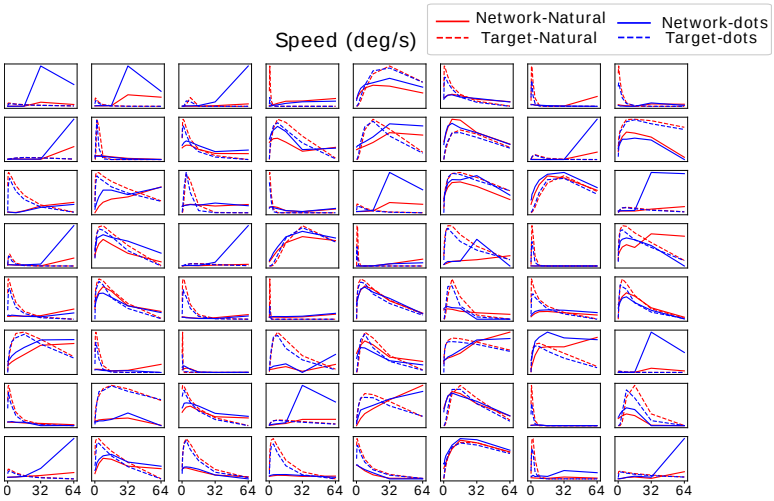


Figure 7.11: Speed-tuning curves of example units in CNN-OMT. Conventions as in Figure 7.8. Mean ± SD for correlation between natural-scene stimulus tuning of target (dashed red) and network units (solid red): 0.20 ± 0.55, and for correlation between dot stimulus tuning of target (dashed blue) and network units (solid blue): 0.27 ± 0.54.

the MT representation is compatible with the odometry task. I also trained an odometry network with video input, and included another cost term that encouraged an intermediate layer to approximate the empirical model. In this case the task performance was barely affected, but the network failed to learn an MT-like intermediate representation. I believe this is a robust negative result. When I first trained networks with the MT cost alone, the representation degenerated with further training on the odometry cost (Figure 7.13). When I trained with both costs together, and then continued training without the odometry cost, the MT approximation rapidly improved (Figure 7.4), while the odometry cost went up substantially. Combining the costs did not have a linear effect on unit tuning. For example, compared to CNN-MT, direction tuning was narrower in CNN-O, but wider in CNN-OMT (Figure 7.7). In summary, while I found that an MT-like representation supports the task, I was unable to produce a convolutional network that had both an MT-like internal representation and good task performance. It may be that a different training strategy is needed, or that more physiologically realistic mechanisms are needed earlier in the network,

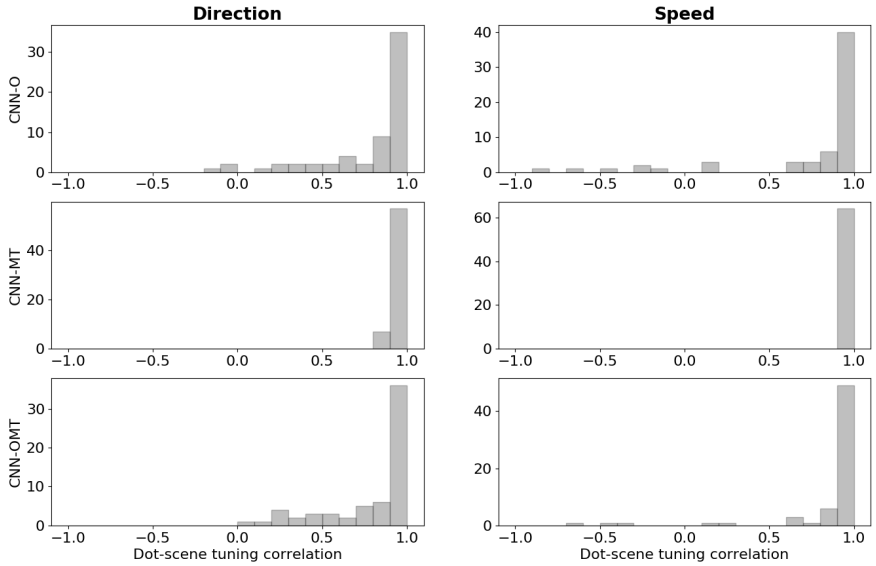


Figure 7.12: Correlations between tuning curves with dot stimuli and scene stimuli. Correlations for direction-tuning curves are shown on the left and those for speed-tuning curves are shown on the right. The correlations are frequently very high in the CNN-MT network (middle). However, there are many uncorrelated cases in the CNN-O network (top), and the CNN-OMT network (bottom), indicating that tuning in these networks is sensitive to details of the stimuli.

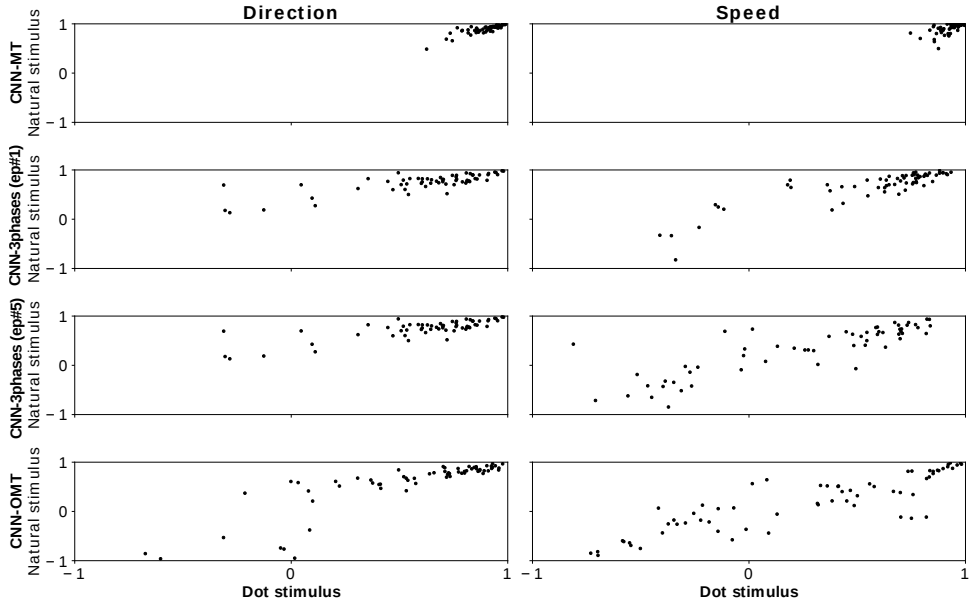


Figure 7.13: Correlations between empirical-model tuning curves and CNN tuning curves. Each point indicates these correlations for dot stimuli (horizontal axis) and natural-scene stimuli (vertical axis). Higher correlations mean that the tuning curves more closely reflect the empirical model. This is related to the regression error, but it differs due to the very different distributions of tuning-curve stimuli vs. training stimuli (for example, most training stimuli are not at the units' preferred speed or direction). The top row shows correlations for CNN-MT. Individual units' correlations are similar for dot and natural-scene stimuli. The second and third rows respectively show correlations between empirical-model and network tuning curves in CNN-3Phases after the first and fifth training epochs of the third training phase (i.e., training only for odometry task). As the third training phase progresses, the correlations for many units become weaker. The bottom row shows correlations between empirical-model and network tuning curves in CNN-OMT. Correlations in direction tuning, especially in response to natural scenes, include many high values. However speed-tuning correlations are more spread out. Comparing the two bottom rows demonstrates how the two approaches of training on both MT activity and odometry affects the correlations. The CNN-OMT direction-tuning curves are less similar to those of the empirical model, especially in response to natural scenes, whereas the CNN-3Phases speed-tuning curves are more different from the empirical model, especially in response to dot stimuli.

such as those in [13]. A deeper network that accounts for the visual processing areas before V1 (i.e., the retina and the lateral geniculate nucleus of the thalamus) may be needed as they were not considered in the present network. I also suspect that it is important for the network to perform a realistic range of tasks, rather than just visual odometry. Optimizing the MT representation for any single task may bias the representation toward properties that are useful for that task, rather than making it more realistic.

It would also be useful in future work to explore variations of the CNN-MT network, aiming to maximize similarity between target and actual tuning curves. In addition to standard hyperparameter tuning approaches, other potential avenues include balancing or weighting training data differently (corresponding more closely to tuning curves), using architectures that conform more closely to anatomy [126], and inclusion of more realistic mechanisms such as those in Baker and Bair [13].

Alternative Regression Targets for Deep Representations

Training data for intermediate layers of deep networks could also be obtained directly from neural recordings [10, 225, 135, 151, 106] or from functional magnetic resonance imaging. An ideal neuron-level dataset would include hundreds of neurons, recorded chronically over at least tens of thousands of trials, with a variety of rich visual stimuli. It is not practical to collect such data in the macaque brain, as MT is located deep in a sulcus, preventing use of standard multielectrode arrays without damage to nearby visual areas. So far, recordings with up to 24-electrode arrays have been possible in the macaque [50]. In marmosets, MT is located on the cortical surface, allowing the use of larger electrode arrays [198, 43, 39]. This may allow rich MT activity datasets in the future.

However, my approach has several advantages over potential large-scale MT recordings. One advantage is that the model properties can be modified, allowing investigation of the influence of individual response features on task performance. Also, empirical models allow specification of an attention field at run-time. This should allow generation of attention-modulated activity labels that are consistent with the attention focus of a network, rather

than the (perhaps different and/or unknown) attention focus of the animal. Finally, the model provides infinite labelled data at low cost.

Chapter 8

Conclusion

This thesis has presented an empirical model of MT response statistics. The model was then used for investigating the relationship between two MT tuning features and performance of an ethologically relevant task. The role of MT surrounds in solving two MT-related tasks was also examined. Finally, the model was used to guide deep networks to have more physiologically realistic representation. This chapter summarizes the key contributions of thesis, as well as the possible directions for future work.

8.1 Summary of Contributions

8.1.1 A Novel Model of MT

I developed a video-driven empirical model of activity in the primate middle temporal area (MT). The model draws extensively from the literature on MT response tuning and statistics. It receives arbitrary stereo video as input. It approximates neural data for direction, speed, and disparity tuning better than recent models of MT, as well as reproducing three phenomena not addressed with the previous models. These are local motion integration, change in speed tuning with contrast, and dynamics of pattern selectivity. The model can

be used for generating large populations of synthetic neurons with realistic statistics as extensive modelling of model-parameter distributions has been undertaken. Finally, the proposed model can explain the variance of unseen MT data well.

8.1.2 A Novel Visual Odometry Dataset

Deep neural networks have become the state-of-the-art solutions for a variety of visual tasks where they sometimes surpass human performance. However, training them requires massive amount of data with ground truth, which can be expensive to acquire. A recent trend is to generate synthetic datasets by employing advanced game-development frameworks. Creating such datasets is inexpensive while dataset statistics can be arbitrarily modified. It has been shown that deep networks, which are trained only on synthetic datasets, can perform better on realistic data compared to networks with the identical architecture but trained only on realistic datasets with limited statistics [102].

I generated a synthetic stereo visual odometry dataset that resembles an animal navigating through the environment. While I used the dataset to study MT, it can be also used as a dataset to train networks to achieve better odometry performance on realistic data.

8.1.3 Sensitivity Analysis of Direction and Speed Tuning on Odometry

I studied the influence of direction- and speed-tuning widths on the accuracy of an odometry task. This investigation found the optimal parameters of the MT speed- and direction-tuning-width distributions for solving odometry (i.e., self-motion estimation). The results also suggested that odometry performance is more sensitive to moderate modulations of speed-tuning widths compared to similar modulations of direction-tuning widths; however, elimination of direction tuning has a considerably higher impact than elimination of speed tuning.

8.1.4 Investigating the Role of Surround in Motion-Related Tasks

To study the role of MT surrounds in solving motion-related tasks, I incorporated the empirical model into deep networks that solved odometry and gesture recognition tasks. I designed and trained deep CNNs to solve the odometry task and LSTM networks to solve the gesture recognition task. I compared task performance and network kernels, corresponding to MT surrounds, between variations of these networks. I also examined the influence of MT surround elimination on suppressive-surround strength of the higher-level convolutional layers of these networks. I found that a fairly large family of MT surround structures can be effective for solving the tasks, and the gradient descent and the brain solutions are strikingly different. However, it will be worthwhile to train a network (with a more dorsal-stream-like architecture) to solve a range of MT-related tasks as opposed to a single one to see whether the gradient descent finds surrounds with similar structure as those of the brain. In this study, I also found that retraining can compensate for the elimination of MT surrounds by introducing stronger suppressive surrounds in the higher-level layers of the deep network.

8.1.5 Guiding Representations in Deep Networks

I designed a deep CNN architecture loosely based on the primate dorsal visual stream. I trained three different networks, with this architecture, on the novel odometry dataset. The first network was trained only for the task, the second one only for emulating MT at one of its intermediate layers, and the third one for both. The first and third networks achieved excellent odometry performance while the third network emulated MT well. I also explored in detail the speed and direction tuning of the units of the intermediate layer, which corresponded to MT, in all networks. All three networks exhibited strong speed and direction tuning in their MT layer, however the tuning in the first network (trained only on the task) was unrealistic, and in the third network (trained on task and MT targets) was weakly related to MT. Despite rather unrealistic tuning in the third network, the regression

cost, corresponding to MT targets, was small. This outcome suggested that changes to the regression cost and/or optimizing on a range of MT-related tasks may be needed to produce realistic tuning.

8.2 Future Work

Using the proposed MT model, it is possible to explore specific relationships between different MT response properties, and a range of realistic tasks. While I studied two tuning features as well as MT surround structures, other response properties of MT such as disparity tuning or contrast sensitivity, and pattern selectivity can be also examined. These investigations can include deep networks or alternative approaches such as the optimal linear decoder method for reconstructing targets, which correspond to an MT-related task.

The proposed model can also be useful to guide CNNs to acquire more MT-like representation. A limitation of the presented work, to create such representation, was the sole usage of the odometry task, which corresponds to self-motion perception. This might bias the representation toward properties that were useful for odometry rather than the broader range of functions supported by MT. MT is involved in a range of tasks such as smooth-pursuit eye movement and motion-based segmentation. So optimizing the same network for several relevant tasks might give more physiologically realistic representation.

Furthermore, a variety of network structures (especially those that are more anatomically realistic) can be explored. More realistic mechanisms can also be incorporated in deep networks (such as those in Baker and Bair [13]), which may help to clarify the mechanisms that produce MT representations.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Daniel L Adams and Semir Zeki. Functional organization of macaque v3 for stereoscopic depth. *Journal of Neurophysiology*, 86(5):2195–2203, 2001.

[3] Hirotugu Akaike. A New Look at the Statistical Model Identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.

[4] Thomas D Albright, Robert Desimone, and Charles G Gross. Columnar organization of directionally selective cells in visual area MT of the macaque. *Journal of neurophysiology*, 51(1):16–31, 1984.

[5] John Allman, Francis Miezin, and EveLynn McGuinness. Direction-and velocity-specific responses from beyond the classical receptive field in the middle temporal visual area (MT). *Perception*, 14(2):105–126, 1985.

[6] John Allman, Francis Miezin, and EveLynn McGuinness. Stimulus specific responses from beyond the classical receptive field: neurophysiological mechanisms for local-global comparisons in visual neurons. *Annual review of neuroscience*, 8(1):407–430, 1985.

[7] John M Allman and Jon H Kaas. A representation of the visual field in the caudal third of the middle temporal gyrus of the owl monkey (aotus trivirgatus). *Brain research*, 31(1):85–105, 1971.

[8] Jose-Manuel Alonso and Yao Chen. Receptive field. *Scholarpedia*, 4(1):5393, 2009.

[9] Mohammad OA Aqel, Mohammad H Marhaban, M Iqbal Saripan, and Napsiah Bt Ismail. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5(1):1897, 2016.

[10] Kuniharu Arai, Edward L. Keller, and Jay A. Edelman. Two-dimensional neural network model of the primate saccadic system. *Neural Networks*, 7(6-7):1115–1135, 1994.

[11] Alireza Bab-Hadiashar and David Suter. Robust optic flow estimation using least median of squares. In *Image Processing, 1996. Proceedings., International Conference on*, volume 1, pages 513–516. IEEE, 1996.

[12] Wyeth Bair and Christof Koch. Temporal precision of spike trains in extrastriate cortex of the behaving macaque monkey. *Neural Computation*, 8(6):1185–1202, aug 1996.

[13] Pamela M. Baker and Wyeth Bair. A Model of Binocular Motion Integration in MT Neurons. *The Journal of Neuroscience*, 36(24):6563–6582, 2016.

[14] Pierre Baldi and W Heiligenberg. How sensory maps could enhance resolution through ordered arrangements of broadly tuned receivers. *Biological cybernetics*, 59(4-5):313–318, 1988.

[15] Horace B Barlow. Possible principles underlying the transformations of sensory messages. 1961.

[16] Horace B Barlow. Single units and sensation: a neuron doctrine for perceptual psychology? *Perception*, 1(4):371–394, 1972.

[17] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM Computing Surveys (CSUR)*, 27(3):433–466, 1995.

[18] Ron Bekkerman, Mikhail Bilenko, and John Langford. Scaling up machine learning: Parallel and distributed approaches. page 401. Cambridge University Press, 2011.

[19] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2): 157–166, 1994.

[20] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

[21] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[22] Christopher M Bishop. Pattern recognition and machine learning. pages 225–290. springer, 2006.

[23] Colin Blakemore, Roger H Carpenter, and Mark A Georgeson. Lateral inhibition between orientation detectors in the human visual system. *Nature*, 1970.

[24] Ali Borji and Laurent Itti. State-of-the-art in visual attention modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):185–207, 2013.

[25] Richard T Born and David C Bradley. Structure and function of visual area MT. *Annual Review of Neuroscience*, 28:157–89, jan 2005.

[26] Richard T Born, Jennifer M Groh, R Zhao, and SJ Lukasewycz. Segregation of object and background motion in visual area MT: effects of microstimulation on eye movements. *Neuron*, 26(3):725–734, 2000.

[27] Pinar Boyraz and Stefan Treue. Misperceptions of speed are accounted for by the responses of neurons in macaque cortical area MT. *Journal of Neurophysiology*, 105 (3):1199–211, mar 2011.

[28] Oliver J Braddick, Justin MD O'Brien, John Wattam-Bell, Janette Atkinson, Tom Hartley, and Robert Turner. Brain areas sensitive to coherent visual motion. *Perception*, 30(1):61–72, 2001.

[29] David C Bradley and Richard A Andersen. Center–surround antagonism based on disparity in primate area MT. *The Journal of Neuroscience*, 18(18):7552–7565, 1998.

[30] Frank Bremmer. Navigation in space–the role of the macaque ventral intraparietal area. *The Journal of physiology*, 566(1):29–35, 2005.

[31] Kenneth H Britten. The middle temporal area: Motion processing and the link to perception. In *The visual neurosciences*, pages 1203–1217. MIT press, 2004.

[32] Kenneth H Britten. Mechanisms of self-motion perception. *Annu. Rev. Neurosci.*, 31:389–410, 2008.

[33] W Michael Brown and Alex Bäcker. Optimal neuronal tuning for finite stimulus spaces. *Neural computation*, 18(7):1511–1526, 2006.

[34] Giedrius T Buracas and Thomas D Albright. Contribution of area MT to perception of three-dimensional shape: a computational study. *Vision research*, 36(6):869–887, 1996.

[35] Santiago A Cadena, George H Denfield, Edgar Y Walker, Leon A Gatys, Andreas S Tolias, Matthias Bethge, and Alexander S Ecker. Deep convolutional models improve predictions of macaque v1 responses to natural images. *bioRxiv*, page 201764, 2017.

[36] Charles F. Cadieu, Ha Hong, Daniel L K Yamins, Nicolas Pinto, Diego Ardila, Ethan A. Solomon, Najib J. Majaj, and James J. DiCarlo. Deep Neural Networks

Rival the Representation of Primate IT Cortex for Core Visual Object Recognition. *PLoS Computational Biology*, 10(12), 2014.

[37] Jason Campbell, Rahul Sukthankar, and Illah Nourbakhsh. Visual odometry using commodity optical flow. 2004.

[38] Matteo Carandini and David J Heeger. Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, 13:51–62, nov 2011.

[39] Tristan A. Chaplin, Benjamin J. Allitt, Maureen A. Hagan, Nicholas S. C. Price, Ramesh Rajan, Marcello G. P. Rosa, and Leo L. Lui. Sensitivity of neurons in the middle temporal area of marmoset monkeys to random dot motion. *Journal of Neurophysiology*, 118(3):1567–1580, 2017.

[40] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[41] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[42] Minggui Chen, Yin Yan, Xiajing Gong, Charles D. Gilbert, Hualou Liang, and Wu Li. Incremental Integration of Global Contours through Interplay between Visual Cortical Areas. *Neuron*, 82(3):682–694, 2014.

[43] Spencer C Chen, John W Morley, and Samuel G Solomon. Spatial precision of population activity in primate area MT. *Journal of neurophysiology*, 114(2):869–878, 2015.

[44] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[45] Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*, 2012.

[46] Carol L Colby and Michael E Goldberg. Space and attention in parietal cortex. *Annual review of neuroscience*, 22(1):319–349, 1999.

[47] Dylan F Cooke, Charlotte SR Taylor, Tirin Moore, and Michael SA Graziano. Complex movements evoked by microstimulation of the ventral intraparietal area. *Proceedings of the National Academy of Sciences*, 100(10):6163–6168, 2003.

[48] Balázs Csanád Csáji. Approximation with artificial neural networks. *Master Thesis, Dept. Science, Eotvos Lorand Univ., Budapest, Hungary*, 2001.

[49] Yuwei Cui, Liu D Liu, Farhan A Khawaja, Christopher C Pack, and Daniel A Butts. Diverse suppressive influences in area MT and selectivity to complex motion features. *Journal of Neuroscience*, 33(42):16715–16728, 2013.

[50] Yuwei Cui, Liu D. Liu, James M. McFarland, Christopher C. Pack, and Daniel A. Butts. Inferring cortical variability from local field potentials. *Journal of Neuroscience*, 36(14):4121–4135, 2016.

[51] T. B. Czuba, Alexander C Huk, Lawrence K Cormack, and Adam Kohn. Area MT Encodes Three-Dimensional Motion. *The Journal of Neuroscience*, 34(47):15522–33, 2014.

[52] Peter Dayan and Laurence F. Abbott. *Theoretical Neuroscience*. MIT Press, 2001.

[53] Russell L De Valois, Duane G Albrecht, and Lisa G Thorell. Spatial frequency selectivity of cells in macaque visual cortex. *Vision research*, 22(5):545–559, 1982.

[54] Russell L De Valois, E William Yund, and Norva Hepler. The orientation and direction selectivity of cells in macaque visual cortex. *Vision research*, 22(5):531–544, 1982.

[55] Gregory C DeAngelis and William T Newsome. Organization of disparity-selective neurons in macaque area MT. *The Journal of neuroscience*, 19(4):1398–1415, 1999.

[56] Gregory C DeAngelis and Takanori Uka. Coding of horizontal disparity and velocity by MT neurons in the alert macaque. *Journal of Neurophysiology*, (2):1094–111, feb 2003.

[57] Simon Denman, Vinod Chandran, and Sridha Sridharan. An adaptive optical flow technique for person tracking systems. *Pattern recognition letters*, 28(10):1232–1239, 2007.

[58] James J DiCarlo and David D Cox. Untangling invariant object recognition. *Trends in cognitive sciences*, 11(8):333–341, 2007.

[59] James J DiCarlo, Davide Zoccolan, and Nicole C Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

[60] Antonia Cinira M Diogo, Juliana G M Soares, Alex Koulakov, Thomas D Albright, and Ricardo Gattass. Electrophysiological imaging of functional architecture in the cortical middle temporal visual area of Cebus apella monkey. *Journal of Neuroscience*, 23(9):3881–3898, 2003.

[61] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[62] Marius Drulea and Sergiu Nedevschi. Motion estimation using the correlation transform. *IEEE Transactions on Image Processing*, 22(8):3260–3270, 2013.

[63] R Dubner and SM Zeki. Response properties and receptive fields of cells in an anatomically defined region of the superior temporal sulcus in the monkey. *Brain research*, 35(2):528–532, 1971.

[64] Charles J Duffy and Robert H Wurtz. Sensitivity of MST neurons to optic flow stimuli. i. a continuum of response selectivity to large-field stimuli. *Journal of Neurophysiology*, 65(6):1329–1345, 1991.

[65] Chris Eliasmith and Charles H Anderson. *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. 2004.

[66] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The Difficulty of Training Deep Architectures and the Effect of Unsupervised Pre-Training. *Aistats*, 5:153–160, 2009.

[67] Christian W Eurich and Helmut Schwegler. Coarse coding: calculation of the resolution achieved by a population of large receptive field neurons. *Biological cybernetics*, 76(5):357–363, 1997.

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

[69] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient belief propagation for early vision. *International journal of computer vision*, 70(1):41–54, 2006.

[70] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72, 2016.

[71] Philipp Fischer, Alexey Dosovitskiy, and Thomas Brox. Descriptor matching with convolutional neural networks: a comparison to sift. *arXiv preprint arXiv:1405.5769*, 2014.

[72] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[73] Apostolos P Georgopoulos, Andrew B Schwartz, and Ronald E Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, 1986.

[74] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[75] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[76] Charles D. Gilbert. The constructive nature of visual processing. In Eric Kandel, James H. Schwartz, and Thomas Jessell, editors, *Principles of Neuroscience*, chapter 25, pages 556–576. McGraw Hill, 2013.

[77] Melvyn A Goodale and A David Milner. Separate visual pathways for perception and action. *Trends in neurosciences*, 15(1):20–25, 1992.

[78] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[79] Michael SA Graziano and Dylan F Cooke. Parieto-frontal interactions, personal space, and defensive behavior. *Neuropsychologia*, 44(6):845–859, 2006.

[80] Umut Güçlü and M. a. J. van Gerven. Deep Neural Networks Reveal a Gradient in the Complexity of Neural Representations across the Ventral Stream. *Journal of Neuroscience*, 35(27):10005–10014, 2015.

[81] Umut Güçlü and Marcel A J van Gerven. Increasingly complex representations of natural movies across the dorsal stream are shared between subjects. *NeuroImage*, 145 Part B:6–13, 2017.

[82] Kaiming He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human level performance on ImageNet classification. In *International Conference on Computer Vision*, pages 1026–1034, 2015.

[83] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[84] Wolfgang Heide, Klaus Kurzidim, and Detlef Kömpf. Deficits of smooth pursuit eye movements after frontal and parietal lesions. *Brain*, 119(6):1951–1969, 1996.

[85] Dan Hendrycks and Kevin Gimpel. Adjusting for dropout variance in batch normalization and weight initialization. 2017.

[86] Christopher A Henry, Siddhartha Joshi, Dajun Xing, Robert M Shapley, and Michael J Hawken. Functional characterization of the extraclassical receptive field in macaque v1: contrast, orientation, and temporal dynamics. *Journal of Neuroscience*, 33(14):6230–6242, 2013.

[87] Hilary W Heuer and Kenneth H Britten. Contrast dependence of response normalization in area MT of the rhesus macaque. *Journal of Neurophysiology*, 88(6):3398–3408, 2002.

[88] Ellen C Hildreth and Shimon Ullman. The measurement of visual motion. 1982.

[89] Geoffrey E Hinton, James L McClelland, David E Rumelhart, et al. *Distributed representations*. 1984.

[90] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814. IEEE, 2005.

[91] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[92] Ha Hong, Daniel L K Yamins, Najib J Majaj, and James J DiCarlo. Explicit information for category-orthogonal object properties increases along the ventral stream. *Nature Neuroscience*, 19(4):613–622, 2016.

[93] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[94] Xin Huang, Thomas D Albright, and Gene R Stoner. Adaptive surround modulation in cortical area MT. *Neuron*, 53(5):761–70, mar 2007.

[95] David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

[96] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1): 106, 1962.

[97] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[98] J Nicholas Hunter and Richard T Born. Stimulus-dependent modulation of suppressive influences in MT. *Journal of Neuroscience*, 31(2):678–686, 2011.

[99] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015.

[100] Laurent Itti and Pierre Baldi. Bayesian surprise attracts human attention. *Vision Research*, 49(10):1295–306, jun 2009.

[101] Hueihan Jhuang, Thomas Serre, Lior Wolf, and Tomaso Poggio. A biologically inspired system for action recognition. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. Ieee, 2007.

[102] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 746–753. IEEE, 2017.

[103] Kai Kang and Xiaogang Wang. Fully convolutional neural networks for crowd segmentation. *arXiv preprint arXiv:1411.4464*, 2014.

[104] N Katsuyama, A Yamashita, K Sawada, T Naganuma, H Sakata, and M Taira. Functional and histological properties of caudal intraparietal area of macaque monkey. *Neuroscience*, 167(1):1–10, 2010.

[105] Seyed Mahdi Khaligh-Razavi and Nikolaus Kriegeskorte. Deep Supervised, but Not Unsupervised, Models May Explain IT Cortical Representation. *PLoS Computational Biology*, 10(11), 2014.

[106] William F. Kindel, Elijah D. Christensen, and Joel Zylberberg. Using deep learning to reveal the neural code for images in primary visual cortex. pages 1–9, 2017.

[107] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[108] David Klindt, Alexander S Ecker, Thomas Euler, and Matthias Bethge. Neural system identification for large populations separating "what" and "where". In *Advances in Neural Information Processing Systems*, pages 3509–3519, 2017.

[109] Adam Kohn and J Anthony Movshon. Adaptation changes the direction tuning of macaque MT neurons. *Nature Neuroscience*, 7(7):764–72, jul 2004.

[110] Kishore Konda and Roland Memisevic. Learning visual odometry with a convolutional network. 2015.

[111] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1–9, 2012.

[112] Norbert Kruger, Peter Janssen, Sinan Kalkan, Markus Lappe, Ales Leonardis, Justus Piater, Antonio J Rodriguez-Sanchez, and Laurenz Wiskott. Deep hierarchies in the

primate visual cortex: What can we learn for computer vision? *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1847–1871, 2013.

[113] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.

[114] Lieven Lagae, Hugo Maes, Steven Raiguel, DK Xiao, and Guy A Orban. Responses of macaque STS neurons to optic flow components: a comparison of areas MT and MST. *Journal of Neurophysiology*, 71(5):1597–1626, 1994.

[115] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE, 2010.

[116] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, 2015.

[117] Honglak Lee, Chaitanya Ekanadham, and AY Ng. Sparse deep belief net model for visual area V2. In *NIPS*, page 8, 2007.

[118] Xiang Li, Shuo Chen, Xiaolin Hu, and Jian Yang. Understanding the disharmony between dropout and batch normalization by variance shift. *arXiv preprint arXiv:1801.05134*, 2018.

[119] Stephen G Lisberger. Visual guidance of smooth-pursuit eye movements: sensation, action, and what happens in between. *Neuron*, 66(4):477–491, 2010.

[120] Jing Liu and William T Newsome. Functional organization of speed tuned neurons in visual area MT. *Journal of Neurophysiology*, 89(1):246–256, 2003.

[121] Jing Liu and William T Newsome. Correlation between speed perception and neural activity in the middle temporal visual area. *Journal of Neuroscience*, 25(3):711–722, 2005.

[122] Liu D Liu, Kenneth D Miller, and Christopher C Pack. A unifying motif for spatial and directional surround suppression. *Journal of Neuroscience*, 38(4):989–999, 2018.

[123] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[124] Brian N Lundstrom, Matthew H Higgs, William J Spain, and Adrienne L Fairhall. Fractional differentiation by neocortical pyramidal neurons. *Nature neuroscience*, 11 (11):1335–1342, 2008.

[125] Najib J Majaj, Matteo Carandini, and J Anthony Movshon. Motion integration by neurons in macaque MT is local, not global. *The Journal of Neuroscience*, 27(2): 366–70, 2007.

[126] Nikola T Markov, MM Ercsey-Ravasz, AR Ribeiro Gomes, Camille Lamy, Loic Magrou, Julien Vezoli, P Misery, A Falchier, R Quilodran, MA Gariel, et al. A weighted and directed interareal connectivity matrix for macaque cerebral cortex. *Cerebral cortex*, 24(1):17–36, 2014.

[127] Henry Markram, Eilif Muller, Srikanth Ramaswamy, MichaelW. Reimann, Marwan Abdellah, CarlosAguado Sanchez, Anastasia Ailamaki, Lidia Alonso-Nanclares, Nicolas Antille, Selim Arsever, GuyAntoineAtenekeng Kahou, ThomasK. Berger, Ahmet Bilgili, Nenad Buncic, Athanassia Chalimourda, Giuseppe Chindemi, Jean-Denis Courcol, Fabien Delalondre, Vincent Delattre, Shaul Druckmann, Raphael Dumusc, James Dynes, Stefan Eilemann, Eyal Gal, MichaelEmiel Gevaert, Jean-Pierre Ghobril, Albert Gidon, JoeW. Graham, Anirudh Gupta, Valentin Haenel, Etay Hay, Thomas Heinis, JuanB. Hernando, Michael Hines, Lida Kanari, Daniel Keller, John Kenyon, Georges Khazen, Yihwa Kim, JamesG. King, Zoltan Kisvarday, Pramod Kumbhar, Sébastien Lasserre, Jean-Vincent LeBé, BrunoR.C. Magalhães, Angel Merchán-Pérez, Julie Meystre, BenjaminRoy Morrice, Jeffrey Muller, Alberto Muñoz-Céspedes, Shruti Muralidhar, Keerthan Muthurasa, Daniel Nachbaur, TaylorH. Newton, Max Nolte, Aleksandr Ovcharenko, Juan Palacios, Luis Pas-

tor, Rodrigo Perin, Rajnish Ranjan, Imad Riachi, José-Rodrigo Rodríguez, JuanLuis Riquelme, Christian Rössert, Konstantinos Sfyrakis, Ying Shi, JulianC. Shillcock, Gilad Silberberg, Ricardo Silva, Farhan Tauheed, Martin Telefont, Maria Toledo-Rodriguez, Thomas Tränkler, Werner VanGeit, JafetVillafranca Díaz, Richard Walker, Yun Wang, StefanoM. Zaninetta, Javier DeFelipe, SeanL. Hill, Idan Segev, and Felix Schürmann. Reconstruction and Simulation of Neocortical Microcircuitry. *Cell*, 163(2):456–492, 2015.

[128] Julio C Martınez-Trujillo and Stefan Treue. Attentional modulation strength in cortical area MT depends on stimulus contrast. *Neuron*, 35(2):365–370, 2002.

[129] Julien Marzat, Yann Dumortier, André Ducrot, et al. Real-time dense and accurate parallel optical flow using cuda. In *7th International Conference WSCG*, 2009.

[130] John H Maunsell and David C 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):1127–1147, 1983.

[131] John H Maunsell and David C Van Essen. Functional properties of neurons in middle temporal visual area of the macaque monkey. ii. binocular interactions and sensitivity to binocular disparity. *Journal of neurophysiology*, 49(5):1148–1167, 1983.

[132] John HR Maunsell and David C Van Essen. Topographic organization of the middle temporal visual area in the macaque monkey: representational biases and the relationship to callosal connections and myeloarchitectonic boundaries. *Journal of Comparative Neurology*, 266(4):535–555, 1987.

[133] Gerrit W Maus, Sarah Weigelt, Romi Nijhawan, and Lars Muckli. Does area v3a predict positions of moving objects? *Frontiers in psychology*, 1:186, 2010.

[134] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation, 2015.

[135] Lane T McIntosh, Niru Maheswaranathan, Aran Nayebi, Surya Ganguli, and Stephen A. Baccus. Deep Learning Models of the Retinal Response to Natural Scenes. *Advances in Neural Information Processing Systems 29 (NIPS)*, (Nips):1–9, 2016.

[136] Markus Meister and Marc Tessier-Lavigne. Low-level visual processing: The retina. In Eric Kandel, James H. Schwartz, and Thomas Jessell, editors, *Principles of Neuroscience*, chapter 26, pages 577–601. McGraw Hill, 2013.

[137] William H Merigan and John HR Maunsell. How parallel are the primate visual pathways? *Annual review of neuroscience*, 16(1):369–402, 1993.

[138] Matthias Minderer, Wenrui Liu, Lazar T Sumanovski, Sebastian Kügler, Fritjof Helmchen, and David J Margolis. Chronic imaging of cortical sensory map dynamics using a genetically encoded calcium indicator. *The Journal of physiology*, 590(1): 99–107, 2012.

[139] Mortimer Mishkin, Leslie G Ungerleider, and Kathleen A Macko. Object vision and spatial vision: two cortical pathways. *Trends in neurosciences*, 6:414–417, 1983.

[140] M Concetta Morrone, DC Burr, and L Maffei. Functional implications of cross-orientation inhibition of cortical visual cells. i. neurophysiological evidence. *Proceedings of the Royal Society of London B: Biological Sciences*, 216(1204):335–354, 1982.

[141] Bernard Moulden, Fred Kingdom, and Linda F Gatley. The standard deviation of luminance as a metric for contrast in random-dot images. *Perception*, 19(1):79–101, 1990.

[142] Anthony J Movshon, Edward Adelson, M Gizzi, and William T Newsome. The analysis of moving visual patterns. In *Pattern Recognition Mechanisms. Eds. Chagas C, Gattass R, Gross C*, volume 54, pages 117–151. Rome:Vatican Press, 1985.

[143] William T Newsome and Edmond B Pare. A selective impairment of motion perception following lesions of the middle temporal visual area (MT). *The Journal of Neuroscience*, 8(6):2201–2211, 1988.

[144] William T Newsome, Robert H Wurtz, MR Dursteler, and Akichika Mikami. Deficits in visual motion processing following ibotenic acid lesions of the middle temporal visual area of the macaque monkey. *The Journal of Neuroscience*, 5(3):825–840, 1985.

[145] William T Newsome, Robert H Wurtz, and Hidehiko Komatsu. Relation of cortical areas MT and MST to pursuit eye movements. ii. differentiation of retinal from extraretinal inputs. *Journal of neurophysiology*, 60(2):604–620, 1988.

[146] Dat Tien Nguyen, Firoj Alam, Ferda Ofli, and Muhammad Imran. Automatic image filtering on social networks using deep learning and perceptual hashing during crises. *arXiv preprint arXiv:1704.02602*, 2017.

[147] M. James Nichols and William T. Newsome. Middle Temporal Visual Area Microstimulation Influences Veridical Judgments of Motion Direction. *The Journal of Neuroscience*, 22(21):9530–9540, 2002.

[148] Shinji Nishimoto and Jack L Gallant. A three-dimensional spatiotemporal receptive field model explains responses of area MT neurons to naturalistic movies. *The Journal of Neuroscience*, 31(41):14551–64, oct 2011.

[149] Harris Nover, Charles H Anderson, and Gregory C DeAngelis. A logarithmic, scale-invariant representation of speed in macaque middle temporal area accounts for speed discrimination performance. *The Journal of Neuroscience*, 25(43):10049–60, oct 2005.

[150] Marie Engelene J Obien, Kosmas Deligkaris, Torsten Bullmann, Douglas J Bakkum, and Urs Frey. Revealing neuronal function through microelectrode array recordings. *Frontiers in neuroscience*, 8:423, 2015.

[151] Michael Oliver and Jack Gallant. A deep convolutional energy model of v4 responses to natural movies. *Journal of Vision*, 16(12):876–876, 2016.

[152] G.A. Orban. Higher order visual processing in Macaque extrastriate cortex. *Physiological Reviews*, 88:59–89, 2008.

[153] Guy A Orban, Lieven Lagae, A Verri, Steven Raiguel, D Xiao, Hugo Maes, and V Torre. First-order analysis of optical flow in monkey brain. *Proceedings of the National Academy of Sciences*, 89(7):2595–2599, 1992.

[154] Christopher C Pack and Richard T Born. Temporal dynamics of a neural solution to the aperture problem in visual area MT of macaque brain. *Nature*, 409(6823): 1040–2, feb 2001.

[155] Christopher C Pack, Vladimir K Berezovskii, and Richard T Born. Dynamic properties of neurons in cortical area MT in alert and anaesthetized macaque monkeys. *Nature*, 414(6866):905, 2001.

[156] Christopher C Pack, J Nicholas Hunter, and Richard T Born. Contrast dependence of suppressive influences in cortical area MT of alert macaque. *Journal of Neurophysiology*, 93(3):1809–1815, 2005.

[157] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[158] Karl Pauwels and Marc M Van Hulle. Optic flow from unstable sequences through local velocity constancy maximization. *Image and Vision Computing*, 27(5):579–587, 2009.

[159] Misha Pavel, George Sperling, Thomas Riedl, and August Vanderbeek. Limits of visual communication: the effect of signal-to-noise ratio on the intelligibility of american sign language. *JOSA A*, 4(12):2355–2365, 1987.

[160] Eli Peli. Contrast in complex images. *Journal of the Optical Society of America. A, Optics and Image Science*, 7(10):2032–2040, 1990.

[161] János a Perge, Bart G Borghuis, Roger J E Bours, Martin J M Lankheet, and Richard J a van Wezel. Temporal dynamics of direction tuning in motion-sensitive macaque area MT. *Journal of neurophysiology*, 93(4):2104–2116, 2005.

[162] John a Perrone and Alexander Thiele. A model of speed tuning in MT neurons. *Vision research*, 42(8):1035–51, apr 2002.

[163] Alon Polsky, Bartlett W Mel, and Jackie Schiller. Computational subunits in thin dendrites of pyramidal cells. 7(6):621–7, jun 2004.

[164] Nicholas J Priebe, Carlos R Cassanello, and Stephen G Lisberger. The neural representation of speed in macaque area MT/V5. *Journal of Neuroscience*, 23(13): 5650–5661, 2003.

[165] Nicholas J Priebe, Stephen G Lisberger, and J Anthony Movshon. Tuning for spatiotemporal frequency and speed in directionally selective neurons of macaque striate cortex. *The Journal of Neuroscience*, 26(11):2941–2950, 2006.

[166] Weichao Qiu and Alan Yuille. Unrealcv: Connecting computer vision to unreal engine. In *European Conference on Computer Vision*, pages 909–916. Springer, 2016.

[167] Steven E Raiguel, MM Hulle, D-K Xiao, VL Marcar, and Guy A Orban. Shape and spatial distribution of receptive fields and antagonistic motion surrounds in the middle temporal area (v5) of the macaque. *European journal of neuroscience*, 7(10): 2064–2082, 1995.

[168] Steven E Raiguel, D-K Xiao, VL Marcar, and Guy A Orban. Response latency of macaque area MT/V5 neurons and its relationship to stimulus parameters. *Journal of Neurophysiology*, 82(4):1944–1956, 1999.

[169] Florian Raudies. Optic flow. 8(7):30724, 2013.

[170] Mengye Ren, Renjie Liao, Raquel Urtasun, Fabian H Sinz, and Richard S Zemel. Normalizing the normalizers: Comparing and extending network normalization schemes. *arXiv preprint arXiv:1611.04520*, 2016.

[171] Mengye Ren, Renjie Liao, Raquel Urtasun, Fabian H. Sinz, and Richard S. Zemel. Normalizing the Normalizers: Comparing and Extending Network Normalization Schemes. pages 1–16, 2017.

[172] Omid Rezai, Pinar Boyraz Jentsch, and Bryan Tripp. A video-driven model of response statistics in the primate middle temporal area. *Neural Networks*, 108:424–444, 2018.

[173] John G Robson. Spatial and temporal contrast-sensitivity functions of the visual system. *Josa*, 56(8):1141–1142, 1966.

[174] Hillary R Rodman and Thomas D Albright. Coding of visual stimulus velocity in area MT of the macaque. *Vision research*, 27(12):2035–2048, 1987.

[175] Hillary R Rodman and Thomas D Albright. Single-unit analysis of pattern-motion selective properties in the middle temporal visual area (MT). *Experimental Brain Research*, 75(1):53–64, 1989.

[176] Constance S Royden. Computing heading in the presence of moving objects: a model that uses motion-opponent operators. *Vision research*, 42(28):3043–3058, 2002.

[177] Daniel B Rubin, Stephen D Van Hooser, and Kenneth D Miller. The stabilized supralinear network: A unifying circuit motif underlying multi-input integration in sensory cortex. *Neuron*, 85(1):1–51, 2015.

[178] Kirsten Rudolph and Tatiana Pasternak. Transient and permanent deficits in motion perception after lesions of cortical areas MT and MST in the macaque monkey. *Cerebral Cortex*, 9(1):90–100, 1999.

[179] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[180] Nicole C Rust, Valerio Mante, Eero P Simoncelli, and J Anthony Movshon. How MT cells analyze the motion of visual patterns. *Nature Neuroscience*, 9(11):1421–31, nov 2006.

[181] Seyed Omid Sadat Rezai. A neurocomputational model of smooth pursuit control to interact with the real world. Master's thesis, University of Waterloo, 2014.

[182] Hide-aki Saito, Masao Yukie, Keiji Tanaka, Kazuo Hikosaka, Yoshiro Fukada, and E Iwai. Integration of direction signals of image motion in the superior temporal sulcus of the macaque monkey. *Journal of Neuroscience*, 6(1):145–157, 1986.

[183] H Sakata, H Shibutani, Y Ito, and K Tsurugai. Parietal cortical neurons responding to rotary movement of visual stimulus in space. *Experimental Brain Research*, 61(3): 658–663, 1986.

[184] Hideo Sakata, Hidetoshi Shibutani, Kenji Kawano, and Thomas L Harrington. Neural mechanisms of space vision in the parietal association cortex of the monkey. *Vision research*, 25(3):453–463, 1985.

[185] Emilio Salinas and LF Abbott. Vector reconstruction from firing rates. *Journal of computational neuroscience*, 1(1-2):89–107, 1994.

[186] C Daniel Salzman, Kenneth H Britten, and William T Newsome. Cortical microstimulation influences perceptual judgements of motion direction. *Nature*, 346(6280): 174, 1990.

[187] C Daniel Salzman, Chieko M Murasugi, Kenneth H Britten, and William T Newsome. Microstimulation in visual area MT: effects on direction discrimination performance. *Journal of Neuroscience*, 12(6):2331–2355, 1992.

[188] Andreas T Schaefer, Matthew E Larkum, Bert Sakmann, and Arnd Roth. Coincidence detection in pyramidal neurons is tuned by their dendritic branching pattern. *Journal of neurophysiology*, 89(6):3143–3154, 2003.

[189] Eyal Seidemann, Allen B Poirson, Brian A Wandell, and William T Newsome. Color signals in area MT of the macaque monkey. *Neuron*, 24(4):911–917, 1999.

[190] Behzad Shahraray and Michael K Brown. Robust depth estimation from optical flow. In *Computer Vision., Second International Conference on*, pages 641–650. IEEE, 1988.

[191] Ralph M Siegel and Heather L Read. Analysis of optic flow in the monkey parietal area 7a. *Cerebral Cortex*, 7(4):327–346, 1997.

[192] Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[193] Eero P Simoncelli and David J Heeger. A model of neuronal responses in visual area MT. *Vision Research*, 38(5):743–761, 1998.

[194] Lawrence C Sincich, Ken F Park, Melville J Wohlgemuth, and Jonathan C Horton. Bypassing v1: a direct geniculate input to area MT. *Nature neuroscience*, 7(10): 1123–1128, 2004.

[195] Adam M Slllito, Kenneth L Grieve, Helen E Jones, Javier Cudeiro, and Justin Davls. Visual cortical mechanisms detecting focal orientation discontinuities. *Nature*, 378 (6556):492, 1995.

[196] Matthew A Smith and Adam Kohn. Spatial and temporal scales of neuronal correlation in primary visual cortex. *The Journal of Neuroscience*, 28(48):12591–603, 2008.

[197] Matthew A Smith, Najib J Majaj, and J Anthony Movshon. Dynamics of motion signaling by neurons in macaque area MT. *Nature Neuroscience*, 8(2):220–8, 2005.

[198] Selina S Solomon, Spencer C Chen, John W Morley, and Samuel G Solomon. Local and global correlations between neurons in the middle temporal area of primate visual cortex. *Cerebral Cortex*, 25(9):3182–3196, 2014.

[199] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15:1929–1958, 2014.

[200] Gene R Stoner and Thomas D Albright. Neural correlates of perceptual motion coherence. *Nature*, 358(6385):412, 1992.

[201] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439. IEEE, 2010.

[202] Markus Svensen and Christopher M Bishop. Pattern recognition and machine learning - solutions to the exercises: Web-edition. page 52, 2009.

[203] Christian Szegedy, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[204] Duje Tadin. Suppressive mechanisms in visual motion processing: From perception to intelligence. *Vision research*, 115:58–70, 2015.

[205] Hiroshi Tamura and Keiji Tanaka. Visual response properties of cells in the ventral and dorsal parts of the macaque inferotemporal cortex. *Cerebral Cortex*, 11(5):384–399, 2001.

[206] Keiji Tanaka, Kazuo Hikosaka, Hide-aki Saito, Masao Yukie, Yoshiro Fukada, and E Iwai. Analysis of local and wide-field movements in the superior temporal visual areas of the macaque monkey. *Journal of Neuroscience*, 6(1):134–144, 1986.

[207] Damien Teney and Martial Hebert. Learning to extract motion from videos in convolutional neural networks. In *Asian Conference on Computer Vision*, pages 412–428. Springer, 2016.

[208] Sebastian Thrun and John J. Leonard. *Simultaneous localization and mapping*. Springer Science & Business Media, 2008.

[209] Roger BH Tootell, Janine D Mendola, Nouchine K Hadjikhani, Patrick J Ledden, Arthur K Liu, John B Reppas, Martin I Sereno, and Anders M Dale. Functional analysis of v3a and related areas in human visual cortex. *Journal of Neuroscience*, 17(18):7060–7078, 1997.

[210] S Treue and JH Maunsell. Attentional modulation of visual motion processing in cortical areas MT and MST. *Nature*, 382(6591):539–541, 1996.

[211] Stefan Treue and J.C. Martínez Trujillo. Feature-based attention influences motion processing gain in macaque visual cortex. *Nature*, 399(575-579), 1999.

[212] Bryan P Tripp. Decorrelation of Spiking Variability and Improved Information Transfer through Feedforward Divisive Normalization. *Neural Computation*, pages 1–27, 2012.

[213] Bryan P Tripp. A convolutional model of the primate middle temporal area. In *ICANN*, 2016.

[214] Bryan P Tripp. Similarities and differences between stimulus tuning in the inferotemporal visual cortex and convolutional networks. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 3551–3560. IEEE, 2017.

[215] Doris Y Tsao, Wim Vanduffel, Yuka Sasaki, Denis Fize, Tamara A Knutsen, Joseph B Mandeville, Lawrence L Wald, Anders M Dale, Bruce R Rosen, David C Van Essen, et al. Stereopsis activates v3a and caudal intraparietal areas in macaques and humans. *Neuron*, 39(3):555–568, 2003.

[216] James M G Tsui, J Nicholas Hunter, Richard T Born, and Christopher C Pack. The role of V1 surround suppression in MT motion integration. *Journal of neurophysiology*, 103(6):3123–38, jun 2010.

[217] DC Van Essen, JHR Maunsell, and JL Bixby. The middle temporal visual area in the macaque: myeloarchitecture, connections, functional properties and topographic organization. *Journal of Comparative Neurology*, 199(3):293–326, 1981.

[218] William E Vinje and Jack L Gallant. Sparse coding and decorrelation in primary visual cortex during natural vision. *Science*, 287(5456):1273–1276, 2000.

[219] Brian A Wandell. *Foundations of vision.* Sinauer Associates, 1995.

[220] Helena X Wang and J Anthony Movshon. Properties of pattern and component direction-selective cells in area MT of the macaque. *Journal of Neurophysiology*, page 74.2/OO9, 2016.

[221] Barry Wark, Brian Nils Lundstrom, and Adrienne Fairhall. Sensory adaptation. *Current Opinion in Neurobiology*, 17(4):423–429, 2007.

[222] William Warren. Optic flow. In Leo M Chalupa and John Simon Werner, editors, *The visual neurosciences*, pages 1247–1259. MIT press, 2004.

[223] D Xiao, Steven Raiguel, V Marcar, and Guy A Orban. The Spatial Distribution of the Antagonistic Surround of MT / V5 Neurons. *Cerebral Cortex*, 7:662–677, 1997.

[224] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML-2015*, 2015.

[225] Daniel L K Yamins, Ha Hong, Charles F Cadieu, Ethan a Solomon, Darren Seibert, and James J Dicarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *PNAS*, may 2014.

[226] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356, 2016.

[227] Xiao-Hu Yu, Guo-An Chen, and Shi-Xin Cheng. Dynamic learning rate optimization of the backpropagation algorithm. *Neural Networks, IEEE Transactions on*, 6(3): 669–677, 1995.

[228] Jure Žbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17:1–32, 2016.

[229] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. *Computer Vision–ECCV 2014*, 8689:818–833, 2014.

[230] Congxuan Zhang, Zhen Chen, Mingrun Wang, Ming Li, and Shaofeng Jiang. Robust non-local tv-l1-optical flow estimation with occlusion detection. *IEEE Transactions on Image Processing*, 26(8):4055–4067, 2017.

[231] HongJiang Zhang, John YA Wang, and Yucel Altunbasak. Content-based video retrieval and compression: A unified solution. In *Image Processing, 1997. Proceedings., International Conference on*, volume 1, pages 13–16. IEEE, 1997.

[232] Kechen Zhang and Terrence J Sejnowski. Neuronal tuning: To sharpen or broaden? *Neural computation*, 11(1):75–84, 1999.