Self-Organizing Neural Visual Models to Learn Feature Detectors and Motion Tracking Behaviour by Exposure to Real-World Data

By:

Arjun Yogeswaran

A thesis submitted in partial fulfillment of the requirements for the degree of

PhD in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

School of Electrical Engineering and Computer Science

University of Ottawa

December 2017

Abstract

Advances in unsupervised learning and deep neural networks have led to increased performance in a number of domains, and to the ability to draw strong comparisons between the biological method of self-organization conducted by the brain and computational mechanisms. This thesis aims to use real-world data to tackle two areas in the domain of computer vision which have biological equivalents: feature detection and motion tracking.

The aforementioned advances have allowed efficient learning of feature representations directly from large sets of unlabeled data instead of using traditional handcrafted features. The first part of this thesis evaluates such representations by comparing regularization and preprocessing methods which incorporate local neighbouring information during training on a single-layer neural network. The networks are trained and tested on the Hollywood2 video dataset, as well as the static CIFAR-10, STL-10, COIL-100, and MNIST image datasets. The induction of topography or simple image blurring via Gaussian filters during training produces better discriminative features as evidenced by the consistent and notable increase in classification results that they produce. In the visual domain, invariant features are desirable such that objects can be classified despite transformations. It is found that most of the compared methods produce more invariant features, however, classification accuracy does not correlate to invariance.

The second, and paramount, contribution of this thesis is a biologically-inspired model to explain the emergence of motion tracking behaviour in early development using unsupervised learning. The model's self-organization is biased by an original concept called retinal constancy, which measures how similar visual contents are between successive frames. In the proposed two-layer deep network, when exposed to real-world video, the first layer learns to encode visual motion, and the second layer learns to relate that motion to gaze movements, which it perceives and creates through bi-directional nodes. This is unique because it uses general machine learning algorithms, and their inherent generative properties, to learn from real-world data. It also implements a biological theory and learns in a fully unsupervised manner. An analysis of its parameters and limitations is conducted, and its tracking performance is evaluated. Results show that this model is able to successfully follow targets in real-world video, despite being trained without supervision on real-world video.

Acknowledgments

I would like to acknowledge everyone who has supported me on this long journey in academia. From very early on in my education, my goal has always been to do a PhD in a field related to computer vision and artificial intelligence, and many people have made this possible.

I want to thank some of my elementary and high school teachers and principals for encouraging me in setting high expectations for myself. In university, I had a few professors who went beyond their duty to help me succeed. I'd also like to thank my committee members of past and present. I would like to give a special thanks to everyone involved in the Quinte Regional Science and Technology Fair as well as the Canada-Wide Science Fair for giving me the foundation to do research and fostering my passion in computing.

Thank you to Dmitry Gorodnichy for giving me my first computer vision job, my first publication, and giving me some credibility on my resume. Thank you to Fredrik Larsson for taking a chance on me, for going out of his way to accommodate me as I worked on my PhD and attended conferences, and for giving me a job where I can do exactly the type of work I've dreamed of doing my whole life.

I want to thank my friends who have always stood by me. They've all been very generous, and often went out of their way to accommodate and support me in my extended-duration student lifestyle.

One of the most important thanks goes to Professor Payeur, who readily accepted me as a Masters student and gave me a very generous opportunity during my PhD to pursue research of my choosing. He has been accommodating of my unusual hours, unusual study habits, and unusual requests with the understanding that I would produce something of value, and I hope I have succeeded in that regard. He has been an ally to me in navigating higher education, and generously read thousands of pages worth of text over the last nine years. Ultimately, he gave me the opportunity to pursue work in this field and he supported me in moving on to a dream job in industry. I cannot thank him enough for these things.

To Ma and Baba, thank you for being supportive and being my parents away from home. Any time I needed anything, they were there for me. With an episode of CSI, some encouragement, and a home-cooked meal, they often helped me get closer to the finish line.

To Koli, I owe you. You've had to put up with a lot; making figures for me, listening to me ramble about my research, enduring my complaints about every single thing, telling me what colour things are, my snoring, and having to sacrifice vacations and other fun things so that I could work on this. Thank you for being supportive, for waiting for me, and for having the patience to put up with all of it. Seriously, thanks.

To Amma and Appa, thank you for *everything*. For pushing me to work hard, study, and achieve things that I'm proud of. My father's dream has been for me to accomplish a PhD, and I'm happy to fulfill that dream for him. My mother's dream has been for me to be happy in whatever I do, and I'm fulfilling that dream every day. I'd like to dedicate this thesis to them.

Finally, I would be remiss if I didn't thank everyone involved in the making of the Robocop and Terminator franchises. Those movies started my passion for computer vision and artificial intelligence at a very young age. If not for them, I probably would've become a dancer.

Table of Contents

Table of Figures

Table of Tables

List of Acronyms

CD	Contrastive Divergence
CNN	Convolutional Neural Network
CRBM	Convolutional Restricted Boltzmann Machine
DBN	Deep Belief Network
DNN	Deep Neural Network
DoG	Difference-of-Gaussians
FEF	Frontal Eye Fields
GRBM	Gated Factored Restricted Boltzmann Machine
ICA	Independent Component Analysis
ISA	Independent Subspace Analysis
MT	Middle Temporal Area
MST	Middle Superior Temporal Area
PCA	Principal Component Analysis
PCD	Persistent Contrastive Divergence
PSD	Predictive Sparse Decomposition
RBM	Restricted Boltzmann Machine
SC	Superior Colliculus
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
TICA	Topographic Independent Component Analysis

Chapter 1 Introduction

The use of computer vision has become prominent in the past few decades since the computational power of personal computing devices has increased and become inexpensive. Computer vision has come to solve several automation problems in many domains, ranging from manufacturing to transportation. However, with this computational power increase and widespread application, the limitations of traditional computer vision methods and architectures have become obvious. There was a time when it was believed that more computational power would solve many of the pitfalls of traditional computer vision techniques and the machine learning mechanisms which supported them. In the modern computing era, this concept has proven to not be the case, and the increase in computational power has revealed that many of the techniques and foundations used have not been able to keep up with the promise of ubiquitous and effective systems which can understand and interpret real-world visual data.

Often, when computer vision is used to solve a problem, it is designed rigidly. In fact, many of the technological building blocks of computer vision, such as feature detectors and segmentation algorithms, are hand-designed in order to get maximum accuracy in a narrow domain. A more elegant goal would be to have one set of general algorithms which can easily adapt without as much influence from human designers. In modern computing, where there is an abundance of data, computational power, and algorithms which are capable of learning, a more robust approach would be to let the algorithms be their own designer, adjusting themselves directly as a result of the data they are trying to model.

1.1 Limitations in Modern Machine Vision

Robotic navigation has long been a test bed for computer vision. The ability of biological organisms to navigate unknown environments using visual data effectively has not been replicated by algorithms to this date. Bulky multi-modal sensors, complex mathematical models, case-based reasoning, path-planning algorithms and human supervision have all been mainstays of the field; a field where most mobile biological organisms with visual sensors can easily navigate. This is not to say that the biological brain does not contain such models and algorithms, but its abstraction and robustness lends credence to the idea that its methods are simpler and far more effective than the more advanced engineering approaches.

Recent advances in 3D sensing and increased funding in the automotive research industry have begun a recent and surprisingly effective trend in self-driving cars. Beginning with systems such as automated parking [1] and collision avoidance [2, 3], and resulting in complex systems such as Google's self-driving car [4], computer vision has permeated the market in a lucrative domain. However, the most effective tools for automated navigation are not traditional computer vision solutions. For Google's driverless car, a detailed map with GPS is the primary method of navigation, so that following road signs is not necessary. 3D laser rangefinder data is used to model the world surrounding the car. It compares that data to prerecorded data of the area so that it can identify non-stationary objects, such as pedestrians, and avoid collisions. Radar is used to gauge speed and distance of surrounding vehicles in fast-moving situations, and ultimately avoid collisions also. The only area which primarily uses computer vision is the system which uses cameras to deal with traffic lights. In the context of human driving, the vision system is primarily responsible for all of the above-mentioned tasks. The reason for using other systems is due to the limitations of computer vision in replicating the success of humans in those fields.

Historically, humans have been well-suited for security applications. Our ability to discriminate and identify faces, recognize suspicious behaviour, and track people in crowds requires intuition, experience, and a complex visual analysis. The best facial recognition systems in literature boast accuracies of 100% on certain datasets [5], however they fail under non-ideal conditions where the lighting is inconsistent, there are shadows, the face has a non-neutral expression, and the face is not within a tolerable range of depth rotations. The human vision system also suffers from degradation in non-ideal conditions, yet it more robustly identifies faces across all of these conditions [6]. Detection of people in a scene and tracking them for surveillance are also limited in their efficacy, with occlusions and low resolution imagery being the biggest detriments to accuracy [7]. Even under ideal conditions, false positive rates and false negative rates achieved by artificial vision systems are significantly worse than human error rates. Some of it can also be attributed to context-dependency and post-processing logic performed by the human brain to improve accuracy beyond the initial guess [8], which is outside the boundaries of most pedestrian detection algorithms.

For industrial applications, computer vision has become very important in domains such as quality control [9] and sorting [10]. Computer vision is a cost-effective solution, since human employment can be expensive, and human fatigue can result in a decrease in speed and an increase in error. Unfortunately, industrial computer vision still cannot always replicate the success of humans due to limitations in visual analysis, especially under uncontrolled circumstances. For example, in fruit harvesting, a human is able to easily identify objects while computer vision techniques fall short [11].

Augmented reality has also come into the forefront due to the dominance of mobile devices on the market, including tablets, smart phones, and augmented reality glasses. Though machine vision is used in many augmented reality applications, it is limited in its ability for widespread usage due to machine vision constraints such as tracking, depth perception, and the computational understanding of visual content [12]. For example, identification technology in augmented reality, such as identifying objects or retrieving a person’s online profile, requires user intervention to be accurate thus limiting the possibility of automation. To provide such applications with further automation, computer vision techniques must be more effective at the general extraction of reliable information from a scene instead of the specific visual cues required for traditionally-designed systems.

Many of these algorithms have been very successful over the past several decades, due to carefully-crafted design, tuning, and a wealth of research. Those techniques have contributed to a very high level of understanding of visual data and methods to retrieve the most valuable information from it. However, to help bridge the gap between the effectiveness of traditional methods and human-level accuracy, it may be time to approach the same problems from a different perspective.

In all of the mentioned domains, the human visual system is still the gold standard when it comes to solving visual tasks; it is compact, operates in real-time, and is able to improve with more exposure to the domain. For these reasons, it is worthwhile to explore these problems from that perspective; though it may be less effective at first, there is promise that it can far exceed expectations as research continues in that direction. Deep learning is one such biologically-inspired mechanism that has gained traction in modern computer vision systems; it started out as being a promising but limited system [13], and now enjoys widespread use in many commercial systems [14, 15]. The exploitation of an abundance of visual data, despite being unlabeled, is a key component in getting these models to develop useful representations. That same property, of using real-world visual data to drive the self-organization of visual models, is the overall theme of this thesis.

1.2 Motivation and Background

In order to improve on the limitations of computer vision, a more roundabout way of designing a solution is employed in that we draw inspiration from an already-effective mechanism which processes visual data: the biological brain. A lot of the civilized world is designed with the human vision system in mind. Urban planning, surveillance systems, and symbolic communication such as text are all suited for humans to visually absorb and analyze. Thus, it makes sense to pursue the design of a system with similar capabilities and robustness to the biological vision system, such that information in the modern world can be interpreted by machines as it is by people, and automation in the modern world can parallel human performance.

Modeling the characteristics of a mature brain after it has learned to interpret the many features of visual data is a lofty task. An easier task to grasp is the development of systems which can learn their own representations, similar to the plasticity of a brain being exposed to visual stimuli after birth during the critical period. In the early stages of brain development, the critical period is the limited time at which the brain has a heightened sensitivity to external stimuli such that it can develop and adapt to said stimuli [16]. The critical period for vision involves rapid modification of the visual cortex based on stimuli from both eyes [17, 18]. This process is dramatically reduced past the early stages of development, and it stands to reason that visual analysis is based upon the brain's self-organization in those early stages. Despite the plasticity of early learning, the visual cortex is rigid enough to tolerate various viewing conditions, disparities between examples of the same class, and affine transforms such as rotation, scale, and translation. The hierarchical approach of the visual cortex allows the reduction of undesired features and allows a general robustness to visual data under non-ideal circumstances [19]. Computational vision can stand to benefit from this type of learning, self-organizing internal representations, visual robustness and the functional interpretation of those representations for tasks such as classification and contextual inference.

Self-organization is the idea that a disordered system adjusts itself via internal interactions such that order emerges. This idea has long been thought of as an explanation for emergent properties in the biological brain. The concept is applied to how the brain learns and adjusts itself, from a relatively random organization, to become capable of coherently representing and interacting with the world, thus reaching some sort of equilibrium. Computationally, these ideas are expressed using unsupervised learning techniques, which start from a random initialization and end up in a very practical and interpretable state. The basis of this type of learning, especially as it is related to brain modeling and vision tasks, is a data-driven statistical analysis. There is a real advantage to this type of learning, because it consists of a very basic mechanism, becomes dependent on the data, and allows complex patterns to emerge with minimal operator interaction or designer bias. The concept of self-organization will be a crucial theme in this thesis.

In the biological brain, information arrives via limited and inexact sensors, actions are carried out by inexact actuators, and only a small amount of the state of the world can be perceived [20]. Due to this uncertainty, an understanding of the world can be believed but cannot be known, and this belief can be considered a function of probabilities based on evidence. Bayesian statistics are concerned with probabilities, most importantly considering the probability of one event occurring given another event. An extension to this is the idea that one's belief about the world should be updated as a function of new evidence compared to previous beliefs. Such an explanation makes a Bayesian probabilistic model an ideal starting point to create systems that understand the real world as well as model mechanisms in the brain. This thesis will focus on the use of probabilistic models, specifically the restricted Boltzmann machine (RBM), as the fundamental mechanism behind the proposed work.

The recent popularization of unsupervised learning and self-organization has occurred through the popularity of deep networks, which are credited with achieving state-of-the-art performance on a variety of challenging tasks. These neural networks are inspired by the neural structure of the brain. Their multi-layered nature bears resemblance to the hierarchical processing that is known to occur in the brain. Their ability to self-organize based on exposure to data is a direct parallel to the learning which occurs in the brain. A specific implementation of a deep network, the deep belief network, can perceive data as well as generate it, which parallels the ability of the brain to perceive information and generate action. This thesis is heavily influenced by these concepts, and is rooted in the very relevant domains of unsupervised learning, self-organization, and deep learning.

Traditional methods such as feature extraction and motion tracking are well-suited to be updated to purely emergent self-organizing models and are likely to produce better results by using the data as a guide instead of designer intuition. Those are the two applications that this thesis will focus on. In traditional machine vision, many individual techniques exist within the domain of feature representation and motion tracking, each of which is able to solve a specific problem such as detecting contours, isolating colour, or identifying motion. For specific goals, such as detecting a certain object, these individual techniques are combined in a rigid manner to specifically address the problem. In addition to the tedious analysis required to determine an appropriate combination of techniques which will extract the desired image information, significant trial and error is required to adjust these techniques and the parameters which govern them. The accuracy of even the most finely-tuned techniques resides well beneath the accuracy of the human vision system and provides no solution to the limited robustness and limited perceptual intelligence of the traditional approach. Compared to the direct approach of designing a system specifically for a task, research shows that the human brain is able to learn from examples, extract many different types of information, and "understand" this data, all with the use of a single robust cognitive vision system. This thesis intends to achieve an increase in robustness and accuracy of feature representation and motion tracking by using a biologically-inspired approach to learn from and interpret complex image data in a manner that draws parallels to the human vision system. Using publications on state-of-the-art cognitive computer vision and prior work done by our research group for geospatial image analysis, the research presented in this thesis is a natural progression.

1.3 Objectives

The main objective of this research is to create and evaluate self-organizing networks and their learned representation of visual information through probabilistic models. This research produces methods which can readily learn from real-world data to self-organize and increase the accuracy, robustness, and data-driven nature of modern computer vision methods by using probabilistic models and mechanisms inspired by those in the biological brain. There are two major sub-projects in this thesis, each of which tackles a different visual problem using that concept. Several of the original contributions of this thesis will be discussed inline, and a comprehensive list will be presented in the conclusion, section 5.2.

1.3.1 Unsupervised Learning of Visual Features with Temporal, Spatial, and Feature-Space Dependence

In the past decade, machine learning has undergone somewhat of a renaissance through the popularization of deep networks. Based on the unsupervised learning capabilities of the RBM, deep networks have put a spotlight on multi-layered networks which are able to learn representations without supervision. RBMs are single-layer networks which learn compact representations of their training data. They accomplish this by attempting to probabilistically represent information such that it can reconstruct similar data to the training data. The difference between training data and the reconstructed data is the primary method of learning in this system. RBMs are layered on top of each other to form a deep network, where each RBM is trained on the outputs of the previous layer, or the original training data in the case of the first layer. This stacked architecture provides layers of internal representation, and provides a very effective starting point for classification. Citing biological-plausibility, statistically-efficient representation ability, computationally efficient training, and the lack of required labeled training data, deep networks have been responsible for recent advances in ubiquitous applications such as speech recognition [21] on mobile devices. For visual data, deep networks have produced state-of-the-art results for image classification in complex domains [22], harnessing the power of large amounts of unlabeled data, such as YouTube videos. As more experiments move the field forward, researchers get a better understanding of how features are learned, how they can be improved to generate a better representation, and why this is the case.

The first part of this thesis aims to investigate those particular questions, and falls squarely into the domain of unsupervised feature learning from visual data. With minimal preprocessing, training RBMs with raw pixel values produce Gabor-like filters which have proven effective feature extractors. These features are known to be statistically-relevant and are physiologically-consistent as they are similar to the Gabor-like features in the visual cortex. Keeping with the idea of self-organization, here, it refers to how the network finds independent features from being trained on real-world data. This thesis aims to improve and investigate the emergence of those features through self-organization. An original contribution is made by evaluating different methods which induce spatial, temporal, and feature-space neighbourhood dependencies for biasing unsupervised learning mechanisms to learn better, more robust features from real-world visual data.

The best image classification results, where at least the initialization of the neural network is done with unsupervised learning, are produced using multi-layered architectures. To appropriately investigate the quality of features learned, their classification ability, and how they relate to the unsupervised algorithm, a single-layer network is more useful such that a deep network architecture's influence can be negated. Modeled after existing comparisons of single-layer networks [23], this part of the thesis attempts to gauge the effect of introducing temporal, spatial, and feature-space dependencies in the production of discriminative features. An important aspect of dealing with feature detectors for image representation is that they must be sufficiently selective such that they only indicate if one particular feature exists. It is also important that they are invariant, such that they can detect that feature under some limited transformations and non-ideal conditions which exist in the real world. Finally, the complete representation must be evaluated to make sure that good features exist which sufficiently cover the variety of visual data. These concepts are considered and discussed, in addition to comparisons of their performances using several metrics including image classification and invariance measurements. Using a basic pooling mechanism and a basic classifier, this first part of the thesis considers different methods of biasing the data and the RBM in order to determine what concepts produce the features best used for image representation.

1.3.2 Self-Organized Motion Tracking by Learning Action and Representation Jointly

The second part of this thesis introduces the concept of combining representation with action and is the paramount original contribution of the thesis. This concept is demonstrated in the domain of motion tracking, though the concept is meant to be extended to other applications. Biologically-inspired methods, especially unsupervised methods for visual processing, have focused on representation; including the first part of this thesis. However, biological systems must both perceive and act on their environment, correlating to requiring both a representation of the world and interaction with that world. The generative properties of some unsupervised techniques make them very suitable candidates for producing actions for the purposes of interacting with the world. The combination of action and perception, linked through learning joint representations, may prove effective both for functionally and as an explanation for how such behaviour may emerge in biological systems such as the brain. It also serves to explore if learning from real-world data is enough to produce sufficient internal representations and create such complex behaviour.

Reinforcement learning, like deep learning, has also found its way from fringe theoretical use into producing state-of-the-art results over an impressive set of challenges. The idea behind reinforcement learning is that a very coarse learning signal, such as telling the model that its response to some stimulus is good or bad, can eventually organize the model to produce the correct response given the stimulus. There is a history of evidence supporting reinforcement learning in biological systems, ranging from Pavlov’s famous experiment [24] to recent explanations of how dopamine functions in the brain [25]. Computationally, reinforcement learning has been responsible for recent breakthroughs in exceeding human-level game-playing performance [26] and image content description [27]. The biological similarities, as well as the concept that a basic learning rule can craft a complex behaviour, make reinforcement learning a strong parallel to the learning used in this thesis’ motion tracking model.

As unsupervised, or weakly supervised, techniques begin to reveal how the brain recognizes objects and what strategies are best to represent visual information through data-driven analysis, there has been a lack of this methodology in another traditional machine vision mainstay: motion tracking. As mentioned, much of the research focuses on the idea that self-organization is useful for representing information. However, this research realizes that action can also be generated through unsupervised learning mechanisms and self-organization to produce certain complex actions. This is accomplished with a basic learning rule which then allows real-world data to drive the learning that ultimately produces the rest of the model’s behaviour. In this thesis, motion tracking and its biological equivalent, smooth pursuit, are used to demonstrate the efficacy of the proposed link between perception and action in a self-organized system. Though it bears some similarities to Friston’s active inference concept [28], this thesis proposes less of a biological simulation and more of an application which deals fully with real-world data for both training and execution.

The second part of this thesis aims to develop a self-organizing motion tracking model. It will learn to understand the relationships between moving a region of interest and the movement of the visual space surrounding it, resulting in the emergence of the model's ability to track that movement without human guidance. This research will be tested on real-world video datasets to demonstrate its efficacy in real-time on cluttered, complex, real-world video. It will also adhere to representing some of the mechanisms found in the brain to perform this task. The novel self-organizing motion tracking model uses several sub-networks: motion discrimination, gaze movements, and motion tracking, forming a deep network. The motion discrimination module learns to encode motion in sequential frames of video. The gaze movements search the visual space for coincident motion, namely the gaze moving at the same vector as the motion in the video, resulting in the receptive field staying the same. The idea of the receptive field staying the same is defined as *retinal constancy*. The motion tracking learns those coincidences, and encodes the relationships between the motion encoding, the gaze, and image constancy in the receptive field. The system becomes conditioned to follow movement with the gaze based on motion encoding instead of explicit visual search. Making full use of deep generative networks, this original concept uses the discriminative properties to learn correlations between motion encoding and gaze movements, while using generative properties to control the gaze such that it matches the motion, providing an elegant link between perception and action. This link is non-trivial as evidenced by retinal constancy being only a single value, thus it provides the learning with no further information than how similar the receptive field is between timesteps, yet is sufficient for motion tracking behaviour to emerge. As a result, it is one of several important contributions presented in this thesis.

The proposed architecture will demonstrate the feasibility of learning motion features as well as the ability to track motion in a fully unsupervised deep network. Additionally, this will be done using real-world data, to show the ability of this network to learn all of the required associations entirely from real-world stimuli. This thesis, through its original contributions in combining perception and action through discriminative representation and generative action, provides a framework for the biologically-inspired unsupervised learning of motion tracking behavior and self-organization that is required for a cognitive vision system. Modern biologically-inspired artificial intelligence is still at an early stage and this research will play a role in its advancement.

This methodology can be used for autonomous navigation, as well as visual surveillance, military target detection and tracking, medical image analysis, and many other applications. Computer vision is important for computerized systems to autonomously interact with the real world, and the proposed research aims to provide robust and reliable results. A primary focus of this research is to minimize the influence of a model's designer by allowing the model to learn on its own from real-world data, to reduce assumptions and address the limitations produced by hardcoded machine vision constraints. The usefulness of this research also lies in understanding and imitating selected aspects of the human visual system, demonstrating the practicality of cognitive vision approaches, and designing self-organizing, perceptually intelligent image analysis systems to further the field of computer vision.

1.4 Structure of the Thesis

This thesis is divided into 5 chapters. Chapter 1 is the introduction which positions this research among applications in the field of study by addressing limitations in modern machine vision, discussing the motivation for the work, and identifying the primary objectives of this research. Chapter 2 conducts a thorough literature review of concepts that this original research builds upon, concepts that this research positions itself against, and concepts that put this research into context.

Chapter 3 introduces the first contribution of this research, which studies the unsupervised learning of improved visual feature representations from video. The performance of learning feature representations directly from real-world data when sharing neighbouring information spatially, temporally, and in feature space is measured both quantitatively and qualitatively.

Chapter 4 details the paramount contribution of this research, which is a biologically-inspired model which learns to encode motion from real-world video and uses that encoding to self-organize and track motion. This is made possible by a concept that is introduced in this work and called retinal constancy, which links desirable combinations of gaze movements and motion information for a target to create a similar impression between subsequent samples of the scene. The concept is generalized to promote learning a joint representation of discriminative perception and generative action. It is also demonstrated in the context of motion detection and visual attention. The proposed model is described, trained and tested on real-world video, and compared to physiological mechanisms in this chapter.

Chapter 5 summarizes and emphasizes the importance of this thesis, details its contributions, and evaluates future directions for this work.

Chapter 2 Literature Review

In order to undertake a project on self-organizing biologically-inspired computational vision, several pieces of literature and domains of interest must be explored. Bayesian statistics and their application in simulating brain functions will shine light on how a system can self-organize based on real-world data. Biological vision will provide hints as to how the biological brain processes motion. Machine learning will provide methods to coarsely mimic such mechanisms, and traditional solutions which tackle a similar problem will be compared to evaluate the benefits and detriments of this new direction of research.

2.1 Self-Organization and Bayesian Approaches to Modeling Brain Function

Self-organization is a very popular theory for the emergence of form and function in the brain. It is the process by which order emerges through only local interactions in a disordered system.

There are several examples of this concept found in nature. Fish and birds commonly travel in swarms, which is considered a form of self-organization. They are able to form a group and change direction as a collective, yet individually their only functions are to be attracted to others, align with them, and avoid collisions. Yet, with no leader and no global information transmission, it is simply a set of local interactions that produce this result. Self-organization in molecules takes place in the form of self-assembly, where molecules organize themselves into a specific construct without external guidance. An example is the formation of double-helix DNA through only hydrogen bonding of individual DNA strands. The wisdom of the crowds is a self-organizing phenomenon which suggests that a better decision can be made by aggregating the information in a crowd of non-experts rather than consulting individual experts [29]. A classic example is the jelly bean counting contest, where a jar contains some number of jelly beans. The average of the guesses of the crowd provides a result that is reasonably close to the actual number. Controlling for experts, it is rare than any individual makes a better guess than the average. And even rarer is that the same individual who guesses a closer number can do so again in a different experiment. It is also important to note that when additional information is given to the participants, the biased guesses cause a significant deterioration in the aggregate estimate. This is an example of less information being beneficial to the final result. Influenced by nature, the concept of emergence and self-organization is already at use in robotics [30]. When all robots in a group are programmed with the appropriate simple instructions, a collective intelligence and group behaviour emerges. Self-organizing robots have been able to carry out complex actions which would have otherwise been complicated to accomplish when using global guidance information. Modeled after the biological examples discussed earlier, robots now exist that can navigate through mazes as a swarm and self-assemble from a series of smaller robots.

Self-organization in the brain has been a well-supported research topic [31]. The brain is a complex organ which is capable of understanding complex sensory data and controlling the body to perform complex tasks. These tasks show gradual improvement from birth onwards. Though some development is tied to genetic information, the brain is too complex to be wholly directed by that information. It is possible that the genetic information encodes basic instructions which governs the behaviour of the brain and primes it for further development on its own. Without a clear idea on what global force directs these behaviours, there is a natural tendency to attempt to explain it with self-organization. The idea is that the brain begins at some dynamic disordered state with basic instructions and, through time and exposure to external stimuli, results in an arrangement that brings it to equilibrium with its environment. Many studies exist to support the idea that the neural circuitry that is prevalent in the brain is indeed the result of self-organization. The postulation that sensory information coupled with basic neural instructions, such as Hebbian learning [32], is all that is required to drive the development of the brain makes for a very interesting research topic. It also leads to investigating how to process sensory information such that neural behaviour emerges.

The Bayesian brain hypothesis postulates that the brain represents sensory data probabilistically [33]. Bayesian statistics is a fundamental force behind purely data-driven analyses. Bayesian statistics concerns itself with evidence of the actual state of the world being expressed in terms of a variable belief quantity. With a Bayesian framework, probabilities of events can be calculated, differing based on either the absence or presence of evidence. This type of architecture is extremely valuable when learning representations of the world based on data. Specifically, in the field of machine learning, unsupervised techniques making use of Bayesian probabilities are able to produce efficient and physiologically-consistent internal representations of datasets. There is considerable evidence to say that the brain self-organizes by Bayesian means, and this principle has explained a variety of physiological aspects of the brain, including hierarchical cortical structure and recurrent mechanisms using forward and backward propagation [34].

The *free-energy principle* is a unified theory of the brain, extending the idea of the Bayesian brain, which attempts to explain cortical self-organization through the basic principle of minimizing free energy [35]. This is based on the idea that the brain behaves generatively, in that it can recreate sensory inputs based on its internal understanding of the world. Practically, using Bayesian inference, it learns an internal representation by attempting to adapt itself such that its predictions are accurate and surprises are reduced. Technically speaking, surprise equates to the negative log probability of the sensory input signals, $-\ln(p(\tilde{s}(t))$, where $\tilde{s}(t)$ is the sensory input at time t. Thus, minimizing the long-term average of surprise is the goal.

Minimizing surprise through the model parameters is computationally complex, therefore, a simpler approximation is required. Free energy is a quantity, rooted in information theory, which places boundaries on the evidence within a model of that data. In the case of the brain, sensory inputs are the data. Free energy is always greater than the negative log probability in the data, which is effectively a measure of surprise. An event that is unlikely to occur based on the model is considered a surprise, and the minimization of free energy in the model corresponds to minimizing the likelihood of a surprise by adapting the model. The free energy, also interpreted to be a measure of the model's prediction error, becomes the driving force behind the emergence of internal representation based on sensory input.

Effectively, free energy, defined by Equation (2.1), is a comparison of the real distribution in the world, $q(\Psi)$, and the model's estimated distribution of the world given the sensory data, $p(\Psi|\tilde{s})$, with the Kullback-Leibler divergence, D_{KL}, [36] being the comparison function, where Ψ is the world. The model should update as a function of the state that minimizes the divergence.

$$F = D_{KL}(q(\Psi)||p(\Psi|\tilde{s})) - \ln p(\tilde{s}) \tag{2.1}$$

This equation shows that the free energy represents the difference between the real world distribution and the estimated distribution of the model given the sensory input. It also shows that the free energy will always be greater than the surprise term, since the Kullback-Leibler divergence is non-negative.

Learning comes from adapting unique situations or outliers, since common situations are more predictable and less information-rich compared to the existing representation in the brain. In early stages, everything is unique, while later stages have less frequent outliers. In some ways, the learning network approaches the maximum it can learn as time passes, however another perspective is that the prediction gets so good that it is less likely to change since there are very few situations that cannot be reasonably predicted by the model.

Regarding the biological plausibility of the generative model involved in the free-energy principle, the hierarchical structure in the visual cortex contains forward connections that propagate data upwards from lower cortical areas to higher ones, while backward connections exist that propagate data the opposite way. Typically, forward connections produce responses upwards and backward connections appear to modulate those signals based on top-down information such as context. Such feedback allows two-way communication between cortical areas and is thought to be essential for complex representation, temporal sequence learning, and learning motor control. On a finer scale, dense feedback during learning aids in evolving efficient representations. According to Hinton [37], the goal of the generative model is to learn compact representations while allowing the original input to be reconstructed from those representations. Friston posits that there is merit to the generative model, and that such a model can plausibly simulate biological visual cortical processing with empirical Bayesian learning [38]. There are a number of complexities in the cortical hierarchical model, since it consists of many-to-one mappings, non-linearities, and temporal dependencies. Forward propagation alone is not sufficient to learn in such an architecture, thus feedback connections are required for proper learning and recognition.

Motor control based on generative models is an alternative to the standard modeling of motor control through optimal control theory [39]. As theories of Bayesian learning are applied in other areas of learning, it is a natural progression to consider it for motor control, despite the common use of optimal control theory. Friston formulates optimal control in terms of predictive coding, replaces optimal control with motor neuron reflex arcs, and replaces the cost functions with prior beliefs. This leads to the concept of active inference.

Active inference, based on the free-energy principle, states that both action and perception minimize surprise [35], as opposed to using only perception where sensory data alone drives the learning. It provides a unifying theory which encompasses perception, cognition and action.

The simulation has a world process and a brain model. The world state, $\tilde{\Psi}$, is updated based on the world's current state and actions produced by the brain model, a. Sensory information, $\tilde{s}$, comes into the brain model based on the world' current state as well as actions generated by the brain model. The brain model has internal states, $\tilde{\mu}$, which attempt to model the world state based on prior expectations of the world's dynamics, $\tilde{\eta}$. Its internal states are updated to states which best minimize the free energy based on the previous internal states and the current sensory input. An action which best minimizes the free energy is generated from those internal states as well as sensory information. Random fluctuations are induced as noise in both the world states and the sensory information. These dependencies, which are a reasonable simulation of world-brain interaction, are simplified and shown in Figure 2.1. Some examples of actions that can be simulated by this theory are smooth pursuit and visual attention, both of which are relevant and both of which will be covered in section 2.2.7.

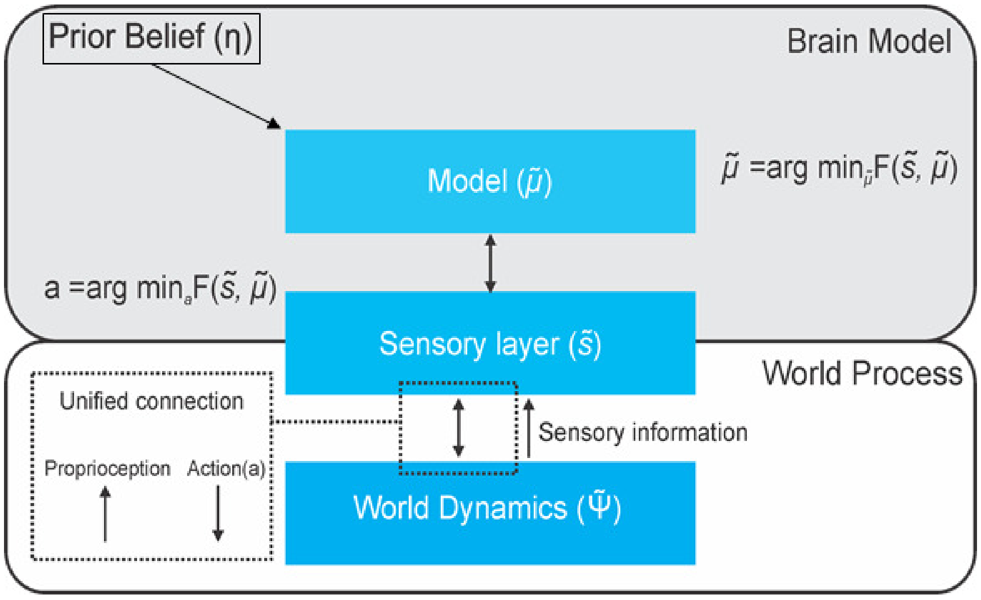

Prior Belief (η)
Brain Model
Model ($\tilde{\mu}$)
$\tilde{\mu}$ =arg min$_{\tilde{\mu}}$F($\tilde{s}$, $\tilde{\mu}$)
a =arg min$_a$F($\tilde{s}$, $\tilde{\mu}$)
Sensory layer ($\tilde{s}$)
World Process
Unified connection
Proprioception
Action(a)
Sensory information
World Dynamics ($\tilde{\Psi}$)

Figure 2.1. Simplified active inference model.

In the figure, $F(\tilde{s},\tilde{\mu})$ is the free energy which the generative recurrent model represents. Internal states are updated based on prior expectations, sensory information, and recurrent connections with the goal of minimizing free energy: $\tilde{\mu} = \tilde{\mu}(t) - \partial_{\tilde{\mu}}F(\tilde{s},\tilde{u})$. Based on these internal states, an automatic generation of actions which maximize the predictive power of the model can proceed: $a(t) = -\partial_a F(\tilde{s},\tilde{u})$. In a sense, the system changes its internal states and its actions to reduce surprise. If the internal states are updated, the system changes its beliefs to match what it is perceiving. If the actions are updated, the system alters the world to match its beliefs. Typically, the updates cause some combination of the two.

The comparison between generated sensations and perceived sensations drives future updates of the model. In implementations of this simulation, the sensory inputs consist of simulation-specific modal sensors as well as action sensors. For example, in a simulation of eye movements based on visual stimuli, the simulation-specific sensors may be visual input while the action sensors report eye position. Based on the physiological concept of the classical reflex arc, where action is generated to reduce error in the action sensors, the brain model generates proprioceptive information which is carried out as action. Proprioception will be further discussed in section 2.2.6.1.

Biologically, this leads to predictive coding and actions interpreted as the prediction error from sensory neurons and motor control by motor neurons. This principle has produced several emergent behaviours consistent with physiological findings, and examples of this model applied to smooth pursuit and saccades are discussed in sections 2.2.6.3 and 2.2.6.4, respectively. A generative model is a practical way to implement such a direct relationship; again the RBM is an example of such a model.

Whether or not self-organization, the free-energy principle, and active inference correspond to how the brain works, they provide a useful principle on which to model emergent behaviour and the link between perception and action through a simple unsupervised mechanism based more on data than rigid rules. This self-organizing unification of action and perception will be the inspiration for the novel motion tracking model introduced in Chapter 4.

2.2 Biological Vision

Before crafting a biologically-inspired solution, it is important to look at some of the architectures of vision in the brain and attempt to find suitable mechanisms which can be adopted into the field of computer vision. A deeper look into the early stages of visual processing, through an investigation of the visual cortex, provides important information into crafting biologically-inspired computer vision algorithms.

2.2.1 Neurons and Synapses

The fundamental component of the nervous system, specifically the brain, is the neuron. Also known as a brain cell, the neuron has a simple function, a very complex architecture, and is still the subject of considerable research on how it behaves. The brain consists of large interconnected networks of neurons, which pass electrochemical signals among themselves to process information and produce action. The average brain consists of over 100 billion neurons, where each one is connected to as many as 10000 other neurons, with over a 100 trillion total connections amongst them. These connections are known as synapses.

In a basic sense, the neuron takes in information from other neurons via electrical signals through the dendrites, then produces its own signal by the axon, which it passes on to other neurons through synapses. Those synapses are connected to the dendrites of other neurons. This process is the most fundamental building block of neural processing. Some neurons are connected to non-neuron cells, often to receive sensory information or to transmit action information. For example, motor neurons output signals through the spinal cord to effectors which produce an action, such as muscle contraction.

The Hebbian learning rule suggests that the strength of the connection between two neurons increases when the excitation of one neuron consistently contributes to the excitation of the other [32]. Thus, neurons that frequently fire together develop a strong connection with each other. This implies that neurons that are strongly connected to each other also provide important information to each other required for crafting a desired function.

The idea of interconnected neurons learning and forming complex functions has produced a plethora of biologically-inspired machine learning techniques, with the artificial neural network being the most popular among them. Most artificial neural network architectures are structured as a directed graph, and have layers of neurons where one layer will propagate information to the next layer. These networks consider the neuron as a simple circuit which takes in weighted inputs from a lower layer of neurons and outputs a transformed signal to the next layer of neurons. Techniques such as backpropagation modify the weights between neurons being modified to produce the desired function. Most techniques use the Hebbian framework such that neurons that contain information important to a connected neuron are weighted strongly; they are weighted positively if one neuron should excite the other neuron, or negatively if one neuron should inhibit the other neuron. These concepts will be discussed when artificial neural networks are discussed in section 2.3.

2.2.2 Vision in the Brain

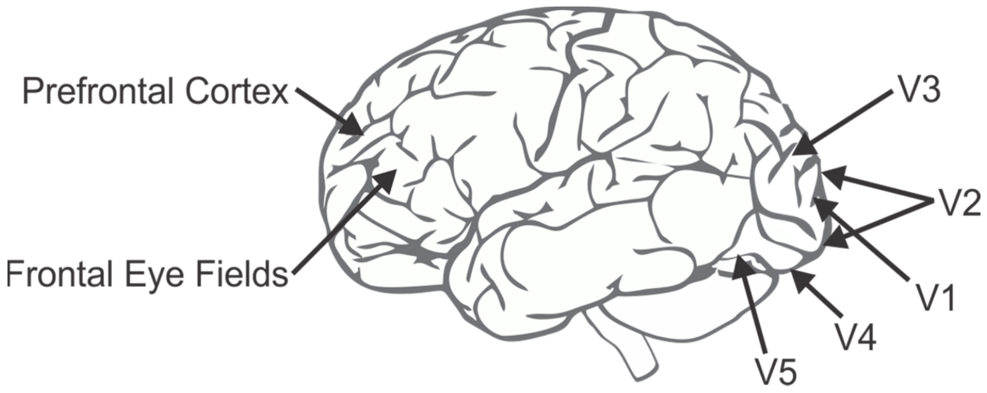


Figure 2.2. Visual cortex in the human brain with certain areas labeled.

The primate visual cortex, shown in Figure 2.2, consists of the primary visual cortex, also known as V1, and the extrastriate visual cortical areas, composed of areas V2, V3, V4, and V5 [40]. V1 processes raw visual input, V2 is used primarily to relay signals to different visual streams, V3 and V4 are both tasked with shape representation, but V4 is also known to process colour information. V5 is primarily associated with motion and the guidance of eye movements, possibly for tracking. An important partner to the visual cortex is the inferior temporal cortex, also known as IT, which is associated with the representation of complex objects, including faces and symbols [41].

This cortex can be found in both hemispheres of the brain. When data passes through the primary visual cortex, it enters one of two streams: the dorsal stream or the ventral stream. The dorsal stream, which transmits data from V1 to V2 to V5, transmits information most associated with motion, object locations, and control of the eyes. This is known as the “where” pathway, since it deals with spatial information and visual action guidance. The stream is essential in perceiving and analyzing spatial relationships and learning to use visual information to coordinate its body in space. The ventral stream, which transmits data from V1 to V2 to V4 to the inferior temporal cortex, transmits information most associated with the purposes of recognition, object representation, and visual memory. This is known as the “what” pathway, since it deals with object recognition and representation. This stream is able to represent the entire visual space through its neurons, and is essential in identification. For this purpose it contains strong connections to both long-term memory and the dorsal stream to integrate spatial and motion information into its tasks.

Neurons in the visual cortex produce a response when visual stimuli appear within their receptive field, which is the spatial region in the visual field which the neuron responds to. Neuronal tuning adjusts the neuron to respond more strongly to specific patterns in that receptive field. In the earlier stages of the visual cortex, the neurons respond to simple patterns such as vertical or horizontal lines, while later stages of the visual cortex has neurons which respond to more complex patterns, such as faces or symbols .

Early responses in V1 neurons have strong tuning to basic stimuli, such as orientation, spatial frequencies, and colour. The functions of the early responses in V1 are known to be similar to oriented Gabor filters [42]. Later responses in V1 indicate that those neurons are more sensitive to global arrangements in a scene, based on recurrent connections from higher level visual cortex areas, such as V4 or IT. It has been found that higher cortical areas can influence responses at earlier stages of the visual cortex. Visual information entering V1 is coded based on local contrast. Some neurons will encode the contrast information, while others encode the colour information involved in the contrast.

V2 neurons have a lot in common with V1 neurons, since they also respond to properties such as spatial frequency, orientation, and colour. It has been found that V2 has some additional functions such as determining whether the stimulus is part of the foreground or background, and binocular disparity for stereoscopic vision. Some neurons in V2 are also tuned to more complex patterns [43], more so than V1, but significantly less than V4. Also, it has been found that V2 is important in the storage of object recognition memory, and the conversion of short-term memory into long-term memory [44].

V4 is one area in the visual stream that shows strong alterations in firing rates due to selective attention, changing firing rates by up to 40% [45]. V4 shares some similarities to V1 in that it has neurons that are selectively tuned to spatial frequency, orientation, and colour. However, unlike V1, it is also tuned for moderately complex objects, such as shapes. Yet, unlike IT, it does not represent significantly complex objects, such as faces or symbols.

Electrophysical properties of neurons in V5, also known as the middle temporal area (MT), show that many are strongly tuned to the speed and direction of moving visual stimuli [46]. Though V1 also contains neurons tuned to speed and direction, V5 contains neurons tuned to speed and direction specific to complex visual features, and integrates local visual motion into the global motion of complex objects.

The frontal eye fields area (FEF) is located in the frontal cortex of the brain, and is responsible for eye movements, showing activation during voluntary saccades and pursuit [47]. The FEF plays a role in visual attention and eye movements. It has a topographic structure and represents saccade targets in retinal space [48].

The superior colliculus (SC) is a multi-layered component of the brain. The earlier layers mostly receive visual information, and they arrive from the visual cortex and the frontal eye fields [49]. With visual data, the neurons in the early layers encode a 2D topographic map of the world relative to retinal coordinates. However, it is split into two, which each half representing half of the visual field. The deeper layers are related to motor control and are capable of driving all types of eye movements, including saccades, when the stimulation is strong enough. The SC produces outputs from both the earlier and deeper areas, which eventually project to areas in the cerebral cortex that control eye movement. For saccades, neurons in the SC are activated at the location of where the saccade will be driven to, based on the retinal-space topographic mapping, while activity everywhere else decreases.

The pulvinar has shown evidence of regulating visual attention [50], and deals with the initiation and regulation of saccadic movement [51]. It is a part of the dorsal stream which handles motion and spatial positioning. Finally, the brainstem connects the brain to the spinal cord, and controls message passing between the brain and body. The pons is a part of the brainstem and contains the nerves responsible for physical eye movement.

Without going in-depth into functional and physical characteristics, this overview should give some explanation to the distinct areas involved in visual processing to ground this thesis in brain anatomy.

2.2.3 Center-Surround Mechanism

Before visual processing in the brain, visual input must be retrieved from light projected onto the retina. Ganglion cells in the retina have receptive fields composed of rods and cones in the eye, and they provide an effective way of detecting contrast. The receptive fields of retinal cells use a center-surround mechanism, composed of a central circle region and a surrounding circle region. Each region is either inhibitive or excitatory when presented with light. Inhibitory surround regions allow the cell to discern local intensity changes. On-center cells are excited by light in the center and inhibited by light in the surrounding region, meaning that it is responsive to light in the center surrounded by darkness. Off-center cells are the opposite, and are excited by light in the surrounding areas and inhibited by light in the center, meaning that it is responsive to darkness in the center surrounded by light. The size of the receptive field dictates the spatial frequency which the cells are sensitive to. Small receptive field sizes are sensitive to fine contrast details, while large receptive field sizes are sensitive to coarse contrast details.

Computationally, the center-surround mechanism is often simulated by using a difference-of-Gaussians (DoG) calculation [52]. The convolution is essentially the subtraction of two Gaussian smoothed images of different filter widths, and can be written as Equation (2.2).

$$DoG\ (x,y) = \frac{1}{\sqrt{2\pi}}\left(\frac{1}{\sigma_1}e^{-(x^2+y^2)/2\sigma_1^2} - \frac{1}{\sigma_2}e^{-(x^2+y^2)/2\sigma_2^2}\right) \tag{2.2}$$

where σ_1 and σ_2 are the Gaussian kernel widths of the center and surround, respectively. When this filter is convolved with an image, the result is an image that shows local contrast based on the selected widths.

A common pre-processing method in deep learning, which also provides a similar transform to the center-surround mechanism, is whitening. This reduces redundancy in the input by making the features less correlated with each other and making the features have the same variance. The first step is centering, which ensures a mean of 0 across the data point by taking the data and removing the mean. The second step is the actual whitening, which transforms the data such that the features are decorrelated from each other. This is accomplished by taking the original data, *x*, and transforming it into $\tilde{x}$ such that its correlation matrix is the identity matrix, $E\{\tilde{x}\tilde{x}^T\} = \mathrm{I}$. This can be accomplished using eigenvalue decomposition of the covariance matrix:

$$E\{xx^T\} = \mathrm{EDE}^T \tag{2.3}$$

where E is the orthogonal matrix of eigenvectors of E{xx^T}, and D is the diagonal matrix of its eigenvalues in the form diag(d_1, d_2, …, d_n). Whitening is now performed by:

$$x_{whitened} = ED^{-\frac{1}{2}}E^Tx \tag{2.4}$$

where $D^{-\frac{1}{2}}$ is performed by taking the negative square root of each element, diag($d_1^{\frac{-1}{2}}$, $d_2^{\frac{-1}{2}}$, …, $d_n^{\frac{-1}{2}}$). This technique has been used in a variety of deep learning algorithms to, theoretically, learn better results. However, techniques such as the RBM are capable of learning higher-order correlations, meaning that the decorrelation is not always necessary. For visual feature learning, it is shown in [53] that whitening does not enable the RBM produce noticeably better results when there are many hidden nodes used.

In addition to center-surround mechanisms for intensity contrast, experiments suggest that fields with the center-surround mechanism are found in the visual cortex for the purpose of motion contrast [54]. Accordingly, cells are responsive to one direction in the center, and then another direction in the surround. Specifically, the study addresses opposing directions, where the surround can be inhibitory or excitatory for the purpose of discerning information about the motion of foreground objects versus background objects. Cells with inhibitory surround will detect motion contrast such as detecting a small target moving among a background with an opposing velocity. Cells with excitatory surround will detect a lack of contrast, which can be beneficial for identifying background motion where all areas move together. More information about the application of this theory into a motion tracking model is discussed in section 2.2.6.3.

2.2.4 Hubel-Wiesel Hierarchical Vision

Hubel and Wiesel discovered the existence of simple and complex cells in V1 [55]. The simple cells have elongated receptive fields, and respond to long features such as bars and edges. The simple cells have sub-regions which exhibit an excitatory response, and other sub-regions which exhibit an inhibitory response. Complex cells occupy much more of V1 than simple cells, and like the simple cells, they also respond only to specific stimuli, yet their receptive field cannot be divided into a region of positive and inhibitive response. Originally, they were considered to be sensitive to a certain stimulus, such as an oriented line, but were invariant to the exact position of that stimulus. Over time, this definition has expanded to encompass more than just positional invariance, as complex cells exhibit invariance to a greater degree than their simple cell counterparts, including translations, rotations, phase shifts, and scaling. The precise configuration of the stimulus is not important, and the neuron will fire as long as the stimulus is within its receptive field. Some complex cells are also known to respond to direction, meaning that motion in a preferred direction will stimulate the cell, whereas it will be unresponsive to motion in other directions. The principle of invariance to transformations is an important part of this thesis when methods of learning invariant features are measured in Chapter 3.

Hubel and Wiesel [55] describe the hierarchical model, shown in Figure 2.3, which has complex cells receiving inputs from a set of simple cells which are sensitive to the same feature in slightly different positions. The complex cell fires at an elevated rate if any of the simple cells respond strongly to their preferred stimulus, achieving translational invariance. Overall, the hierarchical model idea postulates that the receptive field of each cell at one level is composed of inputs from cells at a lower level in the visual system hierarchy. Above the complex cells, are lower-order hypercomplex cells and higher-order hypercomplex cells, and it is suggested that the relationships between those cells are similar to that between the simple and complex cells. Research also shows that this system is more than a feedforward hierarchy, and that cells at one level can also be influenced by feedback from cells at higher levels in the visual system [56]. Functionally, it is thought that this feedback can direct the lower level cells to put more or less emphasis on certain features or areas driven by goals, however, the mechanism by which this process occurs is still an area of active research.

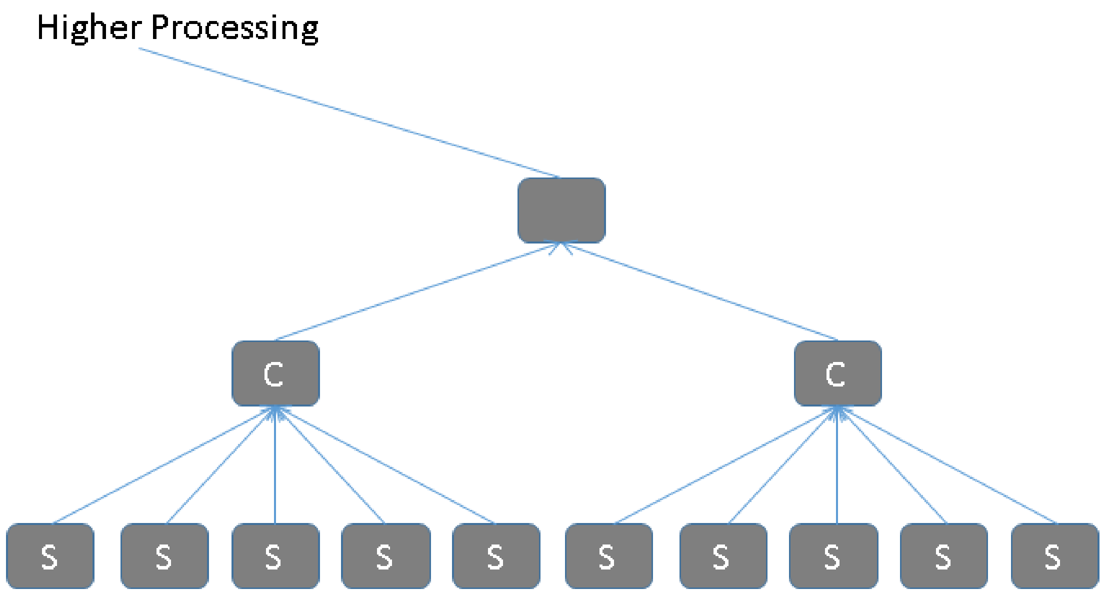


Figure 2.3. Visual hierarchical model composed of simple cells (S), complex cells (C), and higher-order cells (unlabeled).

2.2.5 Motion Discrimination

Motion tracking requires more than just eye movements; it also requires the ability to detect and discriminate between different types of motion.

The dorsal stream in the visual cortex is associated with motion, representation of spatial surroundings, and reflexive control of the eyes and arms when coordinating visual data with body motion. Basic detectors in the visual system have evolved to detect changes in luminance at one position on the retina and correlate it to a change in luminance at another part, thus encoding motion. The Hassenstein-Reichardt detector (also abbreviated as the Reichardt detector) is a simple circuit which is used to model motion-sensitive neurons [57].

This detector measures correlation between two adjacent points, and it consists of two similar subunits. Each subunit receives two brightness values, one directly from its own receptor with a time delay and one from the adjacent receptor with no time delay. The multiplication of these values produce the final output of each subunit. The two subunit outputs are then unified by a subtraction, the order of which his based on the preferred direction. The entire circuit is shown in Figure 2.4. The result of this circuit is a positive response if the brightness moves from the first receptor to the second, a negative response if the brightness moves from the second receptor to the first, and no strong response if both are on, both are off, or the speed at which the movement occurs is not the same as the time delay. This basic detector is the most basic building block of detecting motion, though the specifics of this circuitry in the brain are still under scrutiny. The elaborated Reichardt detector uses spatiotemporal filters instead of light detectors to extend the idea to a more neurologically-plausible and more robust concept [58].

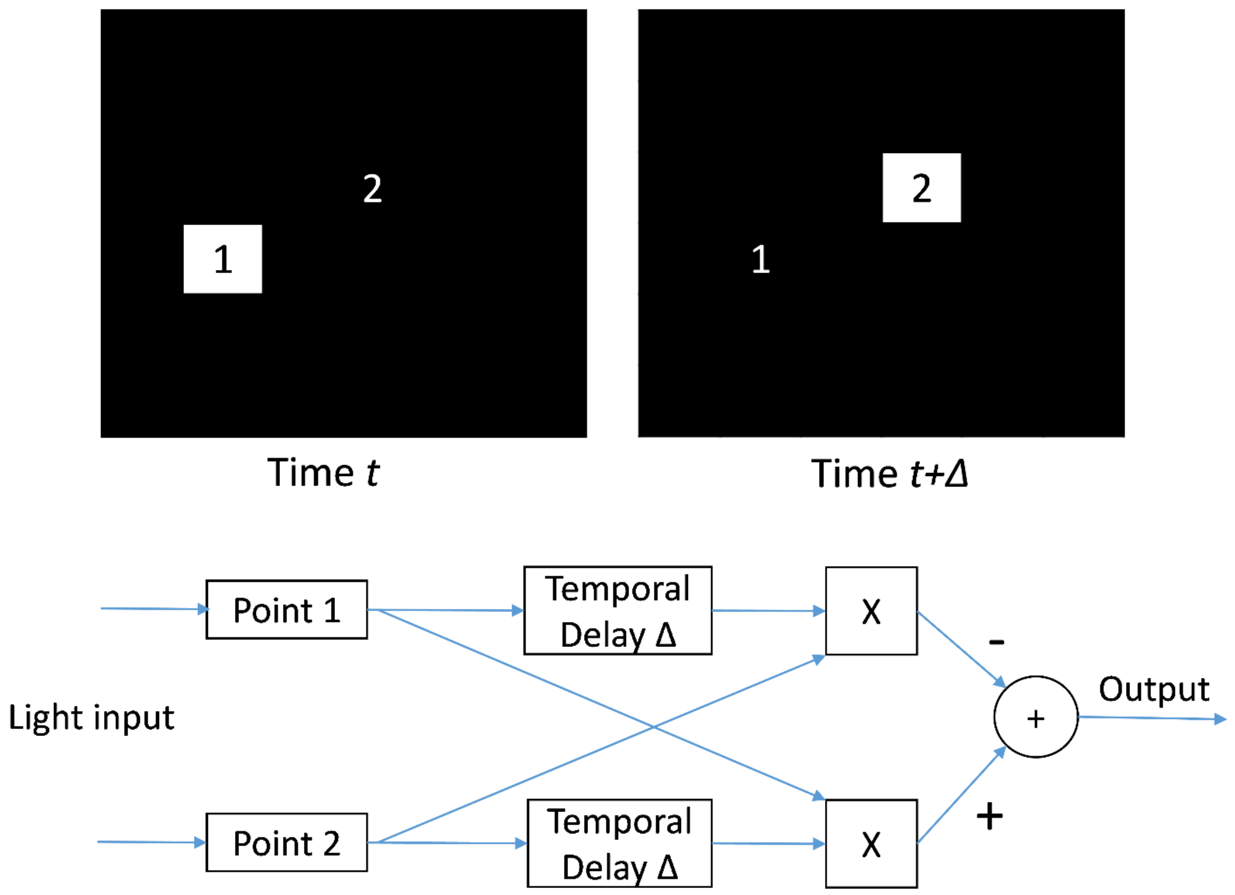


Figure 2.4. Reichardt detector for motion detection. 2D image with the two points being evaluated for motion, where point 1 is lit and point 2 is off, and then after the temporal delay, Δ, point 1 is off and point 2 is lit (Top); Reichardt detector circuit which detects if motion produces a response from point 1 to point 2 over the given temporal delay (Bottom).

Motion-sensitive neurons in V1 respond to motion that occurs locally within their receptive field, with each one being tuned to a certain direction and speed. The integration of different motion-sensitive fields to create a larger understanding of motion occurs in the middle superior temporal area, MST, in conjunction with other areas such as the pons and the cerebellum, and ultimately facilitates the motion tracking via smooth pursuit eye movements.

Wattam-Bell [59] presents evidence that very young infants, between 3 to 5 weeks of age, show no signs of direction discrimination in motion. It seems that direction discrimination emerges when infants are between 6 and 8 weeks of age. However only relative motion seems to appear, and not global motion, which may require higher-order processing based on relative motion analysis. Some theories postulate that such motion discrimination behaviour emerges through self-organization based on exposure to real-world data. For example, such a theory accounts for the difference in performance between horizontal tracking and vertical tracking [60]. Horizontal motion enables better smooth pursuit tracking, as well as more saccades in that direction, as compared to vertical motion. Differences in experience can account for those concepts, meaning horizontal motion is more common than vertical, thus smoother horizontal tracking develops, resulting in increased performance. This thesis, based on the idea that motion detection improves over time in infants, continues on the concept of self-organization even further than motion tracking, to show that motion detection behavior itself can also emerge via self-organization as an encoding of transformation between sequential frames as shown by Memisevic and Hinton [61]. Their work will be further elaborated on in section 2.3.1.3. This physiological perspective is what influences the motion detection layer in the proposed architecture.

2.2.6 Eye Movements

When it comes to perceiving the world, the eyes are often the primary source of information and as a result, a large part of the brain is dedicated to some form of visual processing. However, only *receiving* data limits the visual system to making use of only a small subset of the visual information available. Proactively controlling the movement of the eyes allows an additional level of interaction with the visual data available in the world. This includes the ability to obtain higher-resolution information of a small spatial area [62] and gathering information from a wide range of surrounding areas [63].

There are a several different types of eye movements, all of which serve different purposes and have different methods of activation. Smooth pursuit is the first type of movement, and consists of small movements in order to track a moving object. Saccades are the next type, and are the primary form of changing a point of fixation in the scene, and are characterized by small and large abrupt shifts in gaze. Vergence movements cause both eyes to move in different directions in order to focus on a target based on distance. Finally, vestibule-ocular movements adjust the eyes relative to the external world to compensate for movements of the body and head, such that the eyes can remain fixed on a target despite their local pose.

This thesis will focus mostly on smooth pursuit and saccades.

2.2.6.1 Proprioception

Proprioception is the brain’s sense of stimuli from its body regarding position, motion, equilibrium, and other senses related to movements of the body. It gathers this information from stretch receptors in muscles, tendons, and joints, through sensory neurons located in the inner ear, and through other sensory means. These stretch receptors give an idea of the relative positions of the parts of the body they control, and this information is all integrated in the brain. Though the accuracy of using proprioception alone is high, for more precise tasks, visual input is used as supplementary input for greater accuracy.

However, the accuracy using proprioception alone can be improved with training. Patients that have suffered some injury that damaged their proprioceptive ability have been able to improve with training. Similarly, athletes, musicians, and other people who perform action-dependent activities, can also train their repetitive motions such that it can be done with higher precision without visual input. An example is a concert pianist, who can practice so much that they recite pieces with high accuracy without any visual input.

The eyes also contain proprioceptors, which provide the brain with accurate information about their position. This is required to provide precise spatial positioning of visual input such that the scene can be appropriately mapped to world coordinates based on retinal coordinates and the positioning of the eyes.

2.2.6.2 Reflex Arcs

Reflex arcs are neural pathways which control reflexive actions. Instead of having sensory information pass through the brain and then back to a related reflex, some groups exist which link sensory input neurons and motor neurons through the spinal cord directly. This results in minimal delay between the acquisition of the sensory input and the resulting action. The somatic reflex arc deals with muscular reflex actions, such as the commonly-known knee jerk reflex, where a tap right below the knee causes the lower leg to kick forward. The tap propagates a signal to a sensory neuron which then activates the motor neuron to kick the leg forward, and this is done without any additional processing by the brain.

The vestibulo-ocular reflex is another such time-critical reflex which adjusts the eyes to compensate for head movement, such that the eye remains steady in global coordinates. This serves the purpose of stabilizing the image such that head movement is neutralized and a sharp and consistent image is projected to the center of the visual field. The signal is created by the inner ear, which is able to encode rotation and translation by the head, and propagated relatively directly to the motor neurons controlling the eye muscles [64]. This optimized connection produces one of the fastest reflexes in the human body, and contains only three neurons in the reflex arc to sense head movement and drive eye movement. There are many such reflex arcs which allow an immediate muscular response to sensory input.

A relevant idea to this thesis is that proprioceptive error is reduced by muscle action, which is an important concept in active inference [28, 65]. It considers that proprioceptive error causes an action to reduce that error. From the perspective of the active inference generative model, if proprioceptive error is generated through top-down signals as a prediction, an action to fulfill that prediction will be generated.

2.2.6.3 Smooth Pursuit and Tracking

Smooth pursuit is the process by which the eye tracks a moving object in its field of view [66]. Considered a voluntary process, it is driven by the direction and speed of a stimulus and results in continually shifting the eye such that the moving target always falls approximately in the center of the field of view.

The physiological mechanism that controls smooth pursuit is still the topic of considerable research. Initially, a moving target is seen and the visual data is passed through the primary visual cortex, like any type of vision processing. Motion-sensitive neurons in V1 detect movement with specific direction and speed within their receptive field. The data from the motion-sensitive neurons as well as neurons defining the appearance of the stimulus are propagated through the visual system and end up in V5 where more complex motion information is processed. As discussed earlier, neurons in V5 are selective to the direction and speed of object movement. This motion information is what drives the smooth pursuit process. It is important to note that all of the motion encoded is relative to the retina; however, MST, a neighbouring area to V5, seems to encode object movement in world coordinates as opposed to retinal coordinates.

After the motion information is encoded by V5, higher-level function in the cerebral cortex comes in to play by making the decision on which target to select and whether the target will be pursued or not. The superior colliculus and the frontal pursuit area are thought to be involved in the process to initiate the pursuit movements.

Provided that the target will be tracked, the remainder of the process occurs to generate the appropriate eye movement. The appropriate information is propagated to the pons, an area which controls eye movements as well as a variety of other functions including hearing, taste, and facial expressions. The pons projects to the cerebellum, which contains neurons that represent the target velocity and are responsible for the velocity of pursuit. The cerebellum also corrects for errors in the velocity during the tracking process. Ultimately, the cerebellum propagates signals to the optic motor neurons, controlling the eye muscles and causing the eye to move.

The pursuit itself undergoes two different modes: open-loop and closed-loop. Open-loop is an initial response to the trajectory of the moving object, on the order of 100ms, and more of a rough tracking without any error correction. After this mode, the closed-loop behavior occurs until the pursuit ends. The closed-loop behavior follows the target trajectory more accurately as it constantly corrects for errors in the pursuit speed and direction. Retinal slip is the term used for errors or lags in the position of the eye relative to the target. The online correction attempts to keep the target in the center of the retina such that the object's movement is nullified.

Naturally, higher-level function can modulate the behavior of the smooth pursuit. For example, knowing the trajectory of a stimulus can initiate pursuit before the motion begins and can reduce the amount of retinal slip over the course of the pursuit. In the real world, the target may be occluded for periods of time, and a lot of the pursuit becomes predictive based on the understanding of the trajectory. Thus, higher-level function can strongly aid in the tracking of an object. However, in the absence of a strong understanding of the cognitive functions required, this thesis will focus on the basic parameters of motion tracking.

There is evidence to support that motion tracking behaviour is non-existent at infants under 8 weeks of age through saccadic movements nor smooth pursuit [67]. Smooth pursuit behaviour improves with age up to some limit, as evidenced by the steady progress in tracking accuracy and tracking speed of infants as they mature between 8 and 26 weeks of age [68, 69]. It is possible that the natural maturation process creates this ability, however, the perspective taken in this thesis is that motion tracking behavior can emerge via self-organization based on real-world stimuli. These concepts form the biologically-inspired basis of the motion tracking layer in the proposed architecture.

An earlier paper by Pack *et al.* [70] propose a biologically-consistent model of smooth pursuit in the form of a neural network. They tackle the problem of the eye movement itself complicating the interpretation of motion. When the eye tracks a moving object, the object itself will appear stationary, however the background will move in the opposite direction relative to the retinal motion. If motion detectors are unable to nullify this perceived motion, the eye will oscillate repeatedly going in the opposite direction of the last eye movement. They develop a model which matches many of the physiological properties of smooth pursuit, including many of its limitations. The model is shown in Figure 2.5.

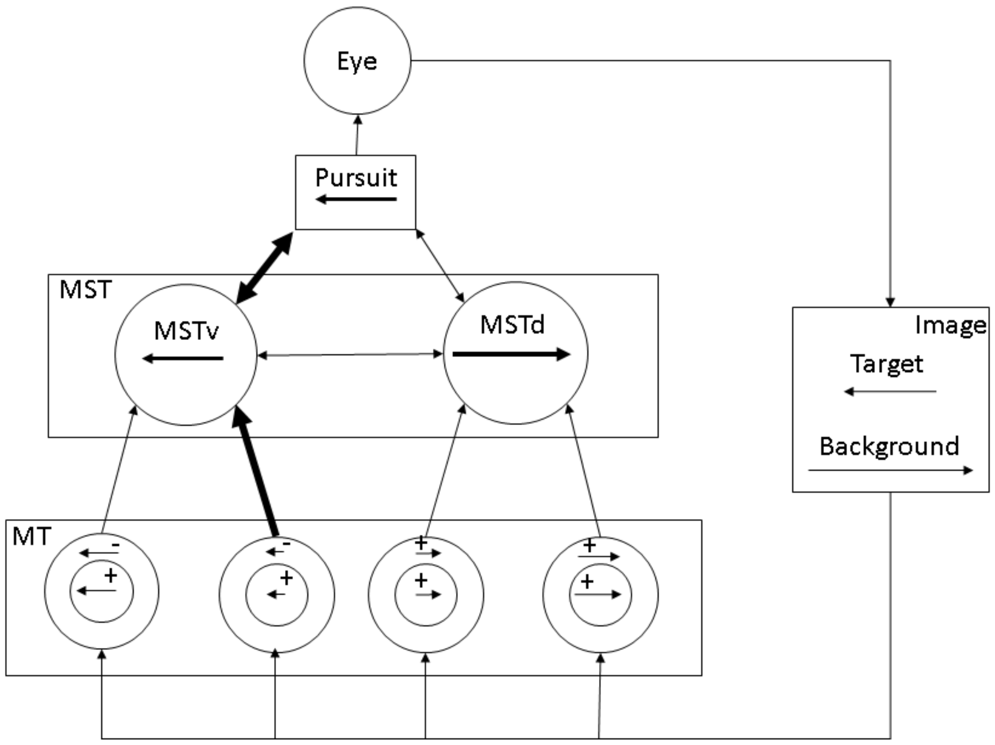


Figure 2.5. The biological model used by Pack *et al.* to explain smooth pursuit. The MT layer contains center-surround motion cells, with excitatory and inhibitory surrounds for similar motion. MST distinguishes between background motion, via MSTd, and foreground motion, via MSTv. The eye then moves relative to the foreground motion encoded by MSTv.

The network has two types of detectors each for left and right motion, representing the MT layer. The first detector has an excitatory surround, such that it fires strongly when the center and the surround present the preferred motion in the same direction, resulting in a strong response when a large area moves in the same direction, such as the background. The other detector has an inhibitory surround, such that it fires strongly when the center is moving in the preferred direction but the surround is not. This results in a strong response when a small area is moving independently from the background, such as a preferred target. Several of these detectors exist for different speeds. These types of detectors have been found in the visual cortex of mammals.

The MST layer has connections from the MT cells and place a higher weight on detectors with larger speeds. The layer contains two types of cells, MSTd and MSTv. MSTd integrates the responses of detectors with excitatory surrounds, meaning it will respond strongly to background motion, and the MSTv integrates the responses of detectors with inhibitory surrounds, meaning it will respond strongly to moving targets. MSTd and MSTv are also laterally connected such that larger background motion supplements smaller target motion when the eye speed begins to approach the target speed. As a result, the cells in MSTv code the predicted global speed of the target, not just the speed relative to the retina. The pursuit layer moves the eye relative to the MSTv code, which represents the target speed, and then propagates its eye movement back down to MSTd and MSTv for future calculations.

The results show the model achieving accurate smooth pursuit over a range of horizontal velocities, and also shows the same limitations found in physiological trials such as slowed pursuit for a target over textured backgrounds. The model simulates smooth pursuit in the visual cortex with custom pre-determined connections and a very specific framework with custom node calculations. Also, it is limited to horizontal motion with hardcoded motion detectors, and does not make use of real-world image or video data. However, it does represent a physiologically detailed simulation with biologically-consistent results.

Shibata *et al.* [71] introduce another biologically-consistent model of smooth pursuit. They advance previous control theory-based models by using a recurrent neural network (RNN) as a model of MST for predicting the current or future velocity. The focus of this paper is on the idea of matching eye velocity to target velocity, and minimizing the retinal slip caused by incorrect prediction or lagging motion. The model is shown in Figure 2.6.

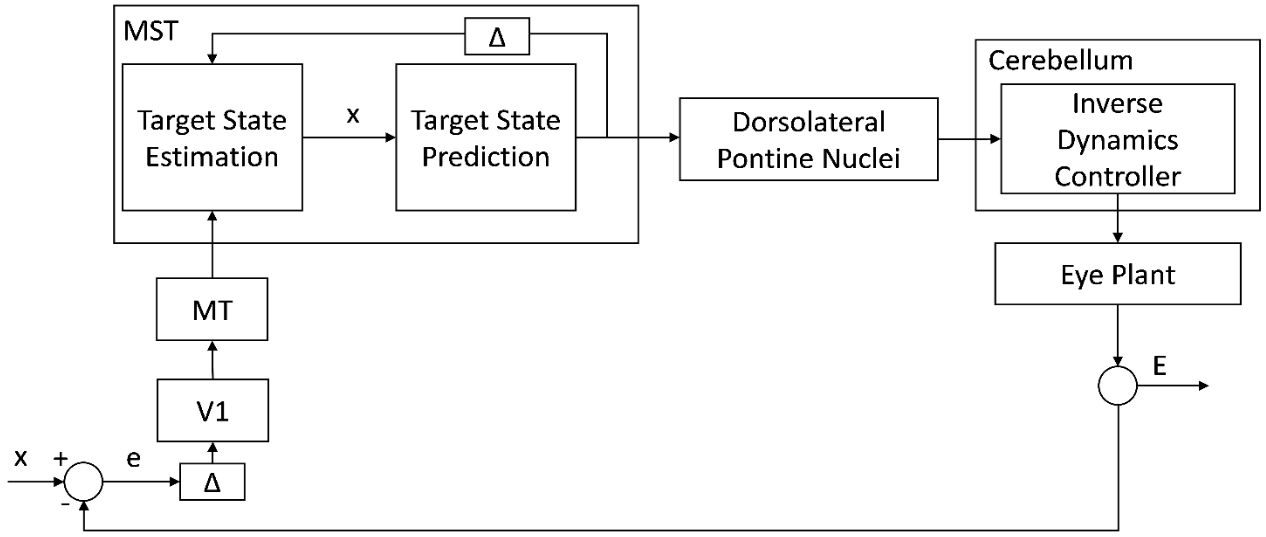


Figure 2.6. Control-theory model of smooth pursuit used by Shibata *et al*.

Like in the previous model, V1 and MT provide the visual input and motion detection components, respectively. However, they do not serve much function in this model other than providing a hardcoded representation of the actual target motion. Here, the MST has been replaced by a mechanism which recurrently estimates the target motion based on previous estimates, using a recurrent neural network. The pursuit and eye control, which occurs in cerebellum, is represented by the inverse dynamics controller which simply moves the eyes.

The framework uses control theory to try match the eye velocity to the target velocity, despite changes in the velocity. Using feedback from the eye velocity and comparing it to the target velocity, the error signal is propagated to the RNN for prediction. This constant comparison allows the RNN to learn the motion of the object and attempt to compensate for retinal slip.

This network does not deal with motion detection, and assumes that the target velocity is known by perfectly accurate motion detectors. However, it does address a very real problem in any tracking application, and solves it using a biologically-inspired recurrent neural network.

2.2.6.4 Saccades

Saccades can be considered sharp abrupt movements of the eye to change the fixation point. Saccades can be both small and large movements, and have been shown to occur both voluntarily and reflexively. Voluntary saccades are simply controlled eye movements, while reflexive saccades occur when the eyes are fixed on a target or even in the process of sleep. Like in smooth pursuit, the superior colliculus is thought to control these saccades [72].

The process of the saccade movement is simple. A target is chosen within the field of view, and the visual system determines the position of the new target with respect to the current target or retinal position. Then, the required eye movement such that the new target falls in the center of the retina is calculated. Like the smooth pursuit mechanism, the information is propagated to the motor neurons which control the eye muscles and thus the eye position. This process takes place during the 200ms delay between the target acquisition and the movement.

Small saccades occur when fixated on a target, and are thought to provide higher resolution detail about a particular target by acquiring snapshots of the target falling on to slightly different retinal positions. Large saccades often occur to acquire more information about a scene and understand the global structure of the view. However, any general quick change of visual fixation can be considered a saccade despite its estimated purpose.

Though it is known that newborns cannot track motion, they can observe by redirecting gaze in a scene through saccadic eye movements [73]. And during the development of motion tracking behaviour, depending on the speed of motion, the eyes indeed alternate between saccadic movements and smooth pursuit movements [68]. The frequent movement is thought to allow the brain to gather more information about visual surroundings, as well as learn better sensory-motor control [74]. When an infant uses a saccade to look at something, and finds that the object has moved by the time it gets there, the brain must learn additional information to produce a better prediction of how to move in the future. This thesis operates on that principle, that the byproduct of getting better at predicting trajectories also facilitates learning to control the eyes to follow that trajectory. The link between saccades and smooth pursuit is an important component of the proposed methodology in this thesis, based on the belief that smooth pursuit can emerge through feedback generated by saccades. The link between tracking processing and the motor control of the eyes is modeled by the relationship between the eye movement layer and the motion tracking layer in the proposed model.

Wang *et al.* [75] use a visual attention model to drive saccadic movements. At each eye movement, they create filter response maps and identify a dynamic saliency map to drive the next fixation point. Comparing their results to human trials, they found that their method predicts the scan path of fixations with a high accuracy.

2.2.7 Self-Organization in Vision

From a biological perspective, there is ongoing debate on whether neural circuitry develops following a predetermined design or whether it can develop as a result of self-organization based on real-world stimuli. The similarities between brain characteristics of different mammals and individuals may indicate that much of its development is predetermined, however, it may also be a natural stable configuration which, over time, is shaped by the data presented to it.

The concept that real-world visual data can cause the emergence of physiologically-consistent results in an unsupervised learning system does not prove that the biological brain functions in a similar manner. However, it does give evidence that real-world visual data might be sufficient for the self-organization of a visual processing system and that efficient representation can emerge via unsupervised learning.

Independent component analysis (ICA) [53] and sparse RBMs [76], which will be further discussed in section 2.3.1, show the emergence of Gabor-like filters similar to simple cells found in V1 when trained with real-world images. Stacked RBMs, further discussed in section 2.3.2, show the emergence of features similar to cells found in V2, which are more complex features resulting from combinations of Gabor-like filters [77]. Independent subspace analysis (ISA) shows the emergence of filters and groupings similar to the invariance properties of complex cells [78]. Topographic independent component analysis (TICA) [79] expands ISA to the idea of producing a 2D topological structure of simple cells similar to the topological structure found in V1 [80]. TICA and ISA will be discussed in section 2.4.3. The gated factored RBM (GRBM) [81] shows the learning of transformation detectors arising from exposure to real-world video sequences, similar to motion detectors in V1. The GRBM will be further discussed in section 2.3.1.3. These properties show that real-world data is quite structured, and both sufficiently rich and sufficiently redundant that very descriptive filters can arise from looking for statistically prevalent features.

Discussed above are more generalized machine learning algorithms, however, many papers discuss perspectives closer to biological simulation. Lücke [82] models a network based on the neural structure of cortical columns found in the visual cortex and uses a custom neuron model which learns based on Hebbian learning. When trained on real-world images, the network produces Gabor-like filters which are more physiologically-consistent than those provided by the more general ICA and sparse coding algorithms.

Bednar and Miikkulainen [83] use a multi-layered self-organizing network with spatiotemporal receptive fields to have a network learn oriented features which are sensitive to motion in a specific direction. Each layer represents a different type of interaction, including photoreceptors for the original image, on-center and off-center surround receptors with different lags to discretely capture motion, and neurons with custom calculations for unsupervised Hebbian learning. Lateral connections help create a topological structure where nearby nodes have similar direction and orientation selectivity.

Regarding smooth pursuit tracking, several papers have tackled the concept of emergence and self-organization. Duran *et al.* [84] use a very basic simulation of the eye and a self-organizing framework known as a coupled chaos system. They do not claim to perform an accurate biological simulation, like other works. They argue that visual saliency is sufficient for the self-organization of a system to track the object. The simulations are performed with a basic black dot on a white background as a target moving at varying speeds along a circular path. The simulated eye measures the position of the object relative to the current position of the center of the eye and uses that information as input to the self-organizing system which eventually tracks the dot by adjusting the eye position. The assumptions made are that visual features can be as simple as a black dot on white background, and that the displacement of the object can be easily calculated. These features are built into the system itself. This system tracks objects just by the act of presenting them to the eye, and does not actually perform any training for long-term self-organization.

Furman and Gur [85] propose a self-organizing network which translates motion from a retinal reference frame to a real-world reference frame. Unlike other methods, which assume a predetermined connectivity of MST units, this network explores the idea of unsupervised learning to develop those connections. The network learns by adjusting the connections between MT and MST, as well as the eye-movement representations, as shown by the dashed arrows in Figure 2.7. The motion detectors, V1, and motion center-surround receptive fields, MT, are hardwired.

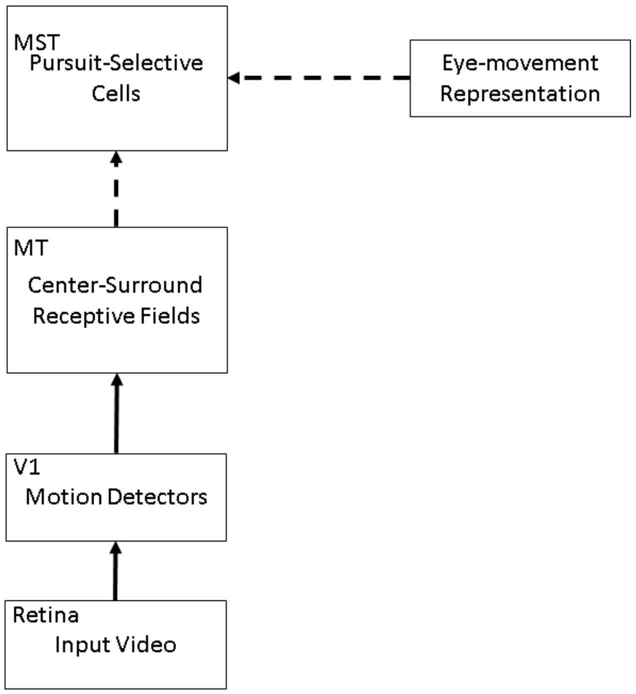


Figure 2.7. Structure of network proposed by Furman and Gur to learn pursuit-selective cells in MST. Dotted lines are learned connections and solid lines are preset connections.

Like in other visual cortex models, the V1 layer provides local motion detectors, while MT contains center-surround mechanisms comparing central motion direction to surrounding motion detection. Some are excitatory, MSTd, and others are inhibitory, MSTv. These center-surround outputs, along with the outputs of eye movements, are what the MST nodes learn from. The network is trained on input movies where pixels are either on or off, containing a small moving target, a moving background, and the associated eye movements during smooth pursuit. The network assumes that smooth pursuit already occurs based on predetermined eye movements associated with the training movies, and the training occurs solely to teach MST units to learn global reference frame motion instead of just the retinal reference frame motion provided by MT.

The results show that each MST unit developed a directional preference during fixation and pursuit. Interestingly, their preferences remain similar whether the eyes were fixated or followed the object. This shows that the MST units are responding independently of the retinal motion to the actual motion of the object. In addition to presenting a biologically-consistent model, this paper shows that the emergent properties consistent with physiological results and self-organization are possible just based on appropriate stimuli.

Friston *et al.* [28, 86] implements the free-energy principle and active inference to study the concept that saccades are used for visual search to test the hypotheses residing in the brain. In this model, saccades are a generated action which collects more surrounding visual information to confirm its beliefs. As the free-energy principle states that the goal of such a model is to reduce unpredictability, if the model knows how its generated actions change sensory data, it can act to reduce surprise. This moves further than predictive coding, which represents current sensory data, and goes into what Friston calls prospective coding, which encodes future sensory data and the actions which acquire them. Friston's model does not learn its priors, but has a predetermined set of images considered as prior beliefs which act as hypotheses as to what the scene could contain. It does, however, recurrently update its internal state to adjust its beliefs about which prior is the correct one to classify the scene it is looking at.

A simple example is given where, if the author is sitting in his garden and notices fluttering in his peripheral vision, his internal brain model will encode the hypothesis that fluttering is caused by a bird. This hypothesis minimizes surprise about the fluttering. Based on that hypothesis, prior beliefs about the gaze are selected such that they minimize uncertainty. Those beliefs produce predictions about the eye movement and the visual stimulation that will occur when the bird is viewed. Based on these beliefs, an action to move the eyes is taken and new visual data after the eye motion is evaluated. If the original hypothesis is correct, maximum evidence will have been gained and the hypothesis can be confirmed with high certainty.

The hierarchical brain model receives only two pieces of sensory input: the proprioceptive input and the visual input. The proprioceptive input provides the system with a noisy signal of where its gaze currently is in the scene. Its visual input, or receptive field, noisily samples a small region of interest at the position of its gaze in the scene. Based on the knowledge of its current position and visual input, the model guesses which prior belief is causing the input. It then generates actions to move the receptive field around to acquire more information to get it closer to confirming its belief. This process repeats, and gathers evidence by obtaining more visual data. After a series of small movements about the last salient position, a saccade is driven to a new area of salience, which is the action that should provide the maximum amount of information among the possible options to help resolve which of its beliefs is correct. It is emphasized that the salience is not a function of the sensory data, but a function of only the internal beliefs generated by that sensory data.

As in the active inference framework described in section 2.1, the simulation is defined by a world process and a brain model. The world process generates the sensory data while the generative model processes the sensory data and generates action.

The world generates sensory data, while the model generates its predicted sensory data based on its existing internal states, or conditional expectations. The noisy sensory data is compared to the model's generated predictions, and the subsequent error signal propagates up the hierarchical model and updates the model's conditional expectations. The generated visual data is a weighted combination of the prior beliefs, produced at s_q. The generated action, *a*, is in the form of the generated proprioceptive signal, which produces a gaze shift that is passed on to the world which affects the next iteration of sensory data. This is a key property, that the same variable that reads the position of the gaze, s_p, also produces the action for the gaze.

The hierarchical generative model contains recurrent connections, *x*, in each layer for temporal continuity. It creates activations, *v*, at each layer based on the recurrent connection and the activations of the layer before it (or the sensory data for the first layer). It acts generatively by producing inputs based on the generated inputs from the layer above it. Error signals are calculated at each layer as some function of the difference between the actual input and the generated input, and those signals are used to update the conditional probabilities.

The process to generate the sensory data consists of proprioception, which reports the position of the center of gaze, and visual input, which provides an image from a region of interest in the full image corresponding to the gaze position, or attentional focus. The entire system is shown in Figure 2.8. Note that action is the only effect that the model has on the world process. Noise and delays are added to more reasonably represent real-world effects and to induce uncertainty to drive updates on belief and action.

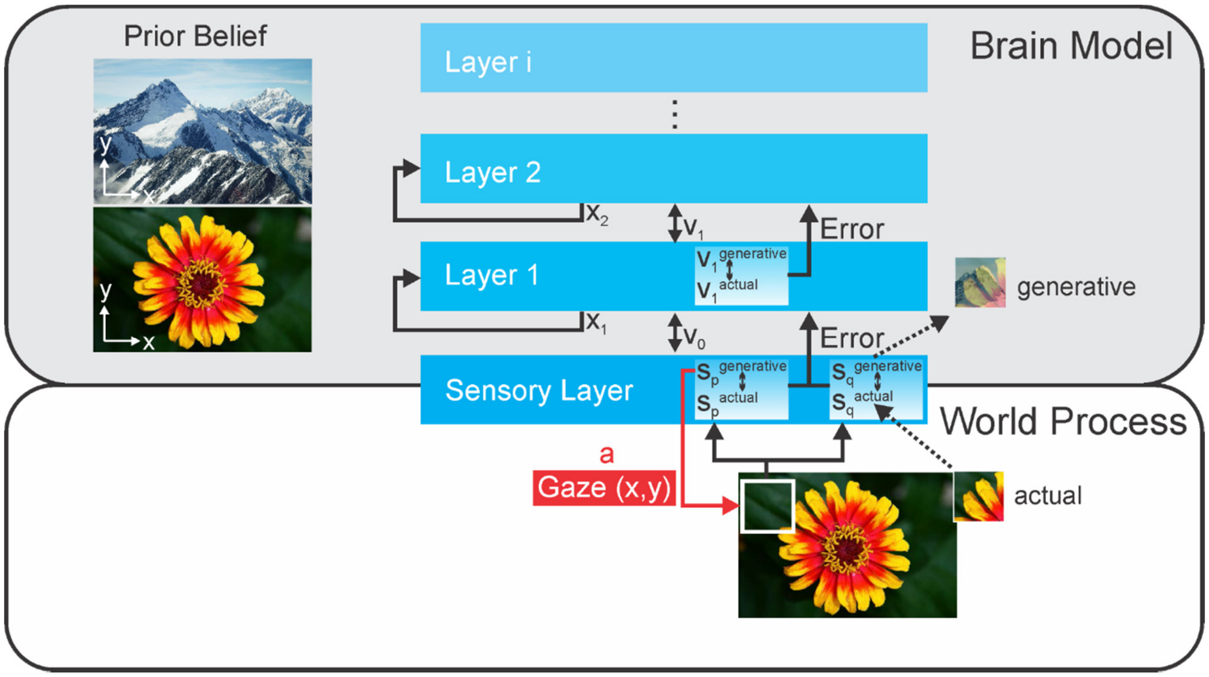


Figure 2.8. Active inference model for visual attention with guided saccades.

In simulations, when presented with an image that is the same as one of the prior beliefs, the model attends salient areas. It also goes to some areas that are less unique, however, this is to gather evidence against the other possible beliefs which have salient features in those areas. However, when it is presented with an image that is not one of the prior beliefs, it struggles to find any valuable information and is not able to isolate one of its prior beliefs as the cause of that scene. This simulation adheres to the idea that small eye movements are used to gather more information about a target, in addition to other physiologically-sound findings.

Each of the mechanisms in Friston's simulation can be represented functionally by areas in the biological brain. Visual input comes through the primary visual cortex and produces prediction errors propagated through the dorsal and ventral streams, which represent "where" and "what" processing, respectively. Internal states in the ventral stream respond to these errors by adjusting their representations to improve predictions based on internal hypotheses. The visual input prediction also depends on the dorsal stream, as the dorsal stream's internal states encode the beliefs about the gaze position. Those beliefs are influenced further by top-down information that identify the direction of gaze that maximizes salience. The salience map, which can be housed by the pulvinar or frontal cortex, is calculated between saccades based on conditional expectations. Beliefs about the gaze direction provides proprioceptive signals to the oculomotor system in the superior colliculus and the pontine nuclei, which produces a proprioceptive prediction error. That error drives the oculomotor system to fulfill beliefs about where to look next. This exact process can be considered as the classical reflex arc, where the action is driven by top-down proprioceptive predictions. This shows that this active inference model can explain the mechanisms in the brain based on an existing understanding of cortical area functions.

Friston's paper is quite important here, as its active inference principles are similar to the principles conveyed by the architecture proposed in this thesis. The concepts of using the same generative framework to update conditional expectations from sensory inputs is also used to generate actions. And the goal of reducing surprise through action, as well as self-organization and emergent behaviour, is a driving force in the proposed architecture. However, the proposed architecture uses more complex real-world data, learns its complex priors from that data, and produces a much more applicable scenario for practical use as opposed to biological simulation on synthetic data.

Adams *et al.* [87] use active inference to model smooth pursuit. The model is very similar to the active inference saccadic model discussed above, however, its internal dynamics cause the action of moving the gaze such that a moving target can be followed. The scene is modeled as a 1-dimensional vector, meaning that position, motion, and sensing only occur along one dimension. As in the saccadic model, there are two sensors: the position sensor and the visual input. However, the representation is slightly different in this model, where the position sensor gives the current angular displacement of the eye in extrinsic coordinates and the visual input provides a binary vector representing where the target is in the receptive field. The receptive field has 17 sensory locations corresponding to the vector, $\vec{r} = [-8, \dots, 0, \dots, +8]$, where 0 is the center of gaze. Thus the receptive field spans 8 positions on either side of the center of gaze. If an element has the value 1, the target is at that position relative to the center of gaze. Otherwise, it will be 0.

The smooth pursuit simulation, as in the saccadic simulation, is defined by the world model and the brain model. The world process creates a target which moves sinusoidally, in world-coordinates, as a function of time. The world process also changes the receptive field position based on the action generated in the brain model.

The brain model creates an attractive prediction towards which the generated target and generated gaze both go. The generated gaze becomes the action that updates the world process. This prediction is a function of internal states which encode the movement of the target and the effect of moving the gaze, and those internal states are then updated by the sensory data. In this simulation, the brain model has prior beliefs of the sinusoidal motion, but its internal dynamics attempt to ascertain the correct parameters of that motion, such as phase and amplitude. Over time, the system adjusts its internal model to the motion of the target and its predictions become more accurate, resulting in it generating actions that closely follow the target by anticipating its sinusoidal movement.

The active inference pursuit system is shown in Figure 2.9.

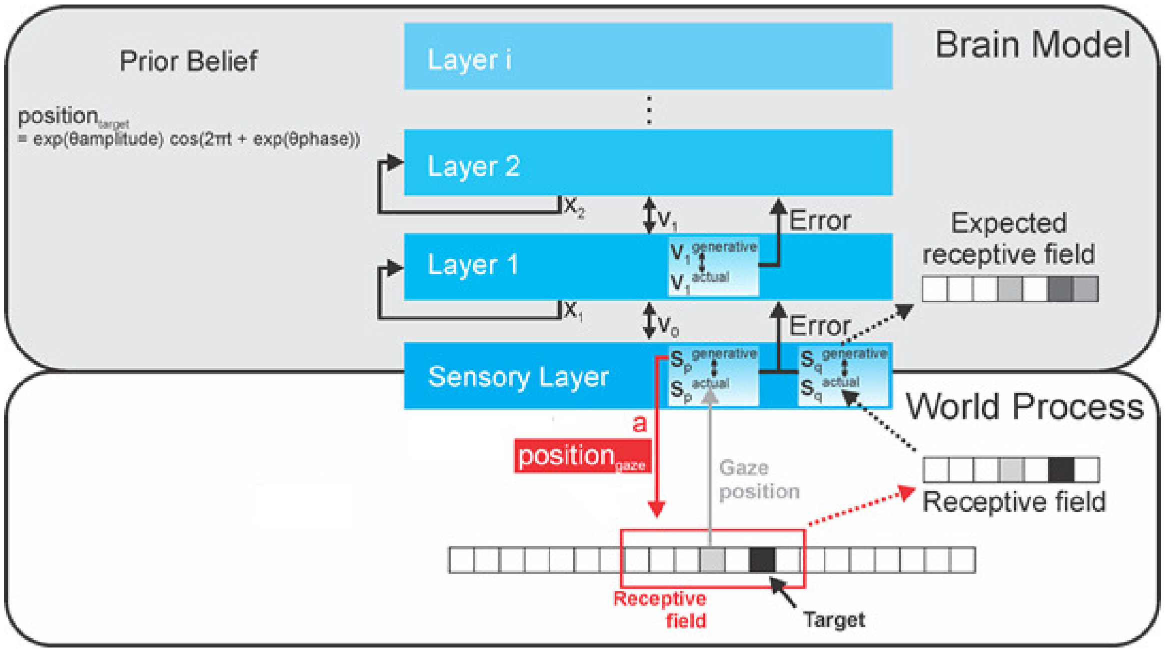


Figure 2.9. Active inference model for smooth pursuit, where the target position moves sinusoidally in the world, and is known by the model, represented internally with prior beliefs. The parameters $\theta_{amplitude}$ and θ_{phase}, represent the amplitude and phase that must be estimated by the model based on the sensory data.

The variables effectively mean the same thing in the pursuit simulation as they do in the saccade simulation. s_p is the proprioception that describes the current position of the gaze with noise added, and *a* is the gaze position that is actually produced. s_q is the receptive field image, which is a small sample of the overall image at the gaze position with noise added. On simulations, when the network is presented with a moving target, it gradually begins to track that motion and predicts its pattern such that pursuit behaviour emerges.

The active inference scheme uses the generative model to great effect by modeling sensory information regarding position and using those same variables to generate action. This is a principle that is also implemented by the proposed motion tracking model in this thesis. However, an important principle implemented in this thesis is that prior beliefs are not hardcoded and are instead learned from the data, which differs from the active inference implementations above. And the learning of these beliefs is important, since the proposed simulations occur on far more complex data than a simple pixel on a 1D vector.

2.2.8 Summary

This section covered the various aspects of the biological visual system. The basic buildings blocks of the brain were discussed, including neurons and the synapses which connect them. The idea that the brain operates as a network, that propagates information modulated by its connections, was established such that it can serve as inspiration for all of the models in this thesis to use neural networks, which are a computational parallel to neural structure. Building upon this, section 2.2, reviewed the basics of how the brain processes visual data.

The center-surround mechanism is the earliest stage of visual processing, where the raw visual data is compressed by removing redundant information and keeping contrast information. Then, the Hubel-Wiesel hierarchical model was reviewed, which theorizes that the visual cortex has several layers that incrementally reduce the dimensionality and spatial resolution of visual data to achieve specificity in recognition and invariance to transformation. Motion discrimination is also important, in that there are neurons that are sensitive to motion. This requires an opposing goal to achieving invariance, in that small changes due to transformation must be accurately encoded for motion.

Moving from perception to action, a survey of the different eye movements produced by the brain was conducted. The eyes control what information is passed to the brain, and the brain controls how the eye moves. This circular dependence is important in this thesis for the development of a self-organizing system that can perceive and control how to perceive. Saccades are the eye movements that infants are born with, and can dart around the scene abruptly to gather information from different areas in the visual space. Pursuit movements move gradually to follow a target. Pursuit develops over time to track motion; as it improves, the brain relies less on saccades to track motion. The relationship between these two eye movements, as well as the gradual emergence of pursuit movements, parallels the model introduced in this thesis, where saccades help the model learn pursuit movements and tracking.

Finally, a discussion of self-organization in vision was presented and some computational models that mimic it were reviewed. The most important of these models is Friston's active inference model, which learns perception and action jointly. Using the principles of active inference, a visual attention model and a tracking model are both described, the latter of which is the most interesting to this thesis.

This overview of the human visual system, and of existing computational models that attempt to mimic them, helps to position the models introduced in this thesis. The latter will apply all of the covered information about visual processing in the brain, and compare themselves to existing literature on computational models which do something similar. The models in this thesis aim at procuring improvements over the existing models both in performance and in understanding about how simpler learning mechanisms and a greater reliance on real-world visual data are sufficient to craft the same behaviour.

2.3 Unsupervised Machine Learning

In the domain of machine learning, there are two major methods of learning: supervised and unsupervised. Supervised machine learning is the process of adjusting a model's parameters based on labeled training data. Each example consists of the input data as well as desired output data, and the learning algorithm attempts to create a mapping from the input data to the output data, and ideally, generalize these relationships to produce the correct output data from future input data.

Feedforward artificial neural networks are among the most popular architectures for supervised machine learning. It is a directed graph composed of layers of artificial neurons, where each of the neurons generate an output as a function of the weighted sum of its inputs coming from the previous layer. The first layer is provided with the input data, and the outputs of each layer are calculated in succession, propagating their values until the top layer is reached and the final outputs are generated. The backpropagation algorithm is the most common supervised training method for the artificial neural network, where the goal of the algorithm is to transform the network into a function which maps the inputs to the outputs with minimal error [88]. The backpropagation algorithm searches for the minimum of the error function in weight space by propagating the training example forward, measuring the error between the produced response and the desired output, and then propagates the error backwards through the network. The weights are updated to bring the error function towards a local minimum through gradient descent.

The support vector machine (SVM) is another supervised learning framework, which attempts to linearly separate the example data into the appropriate classes based on the label data [89]. Since it is unlikely that training examples can be linearly separated by class in their original form, the SVM maps the input data into a higher-dimensional space where this separation can be performed.

Despite the excellent performance of supervised learning frameworks [90], they have significant limitations which can often be addressed by unsupervised learning methods [91]. The most frequent, and obvious, limitation is that many applications have a lack of labeled training data. Often, labeled training data requires laborious manual creation and is often not possible due to time and cost constraints. Along the same line of thought, when looking at biologically-inspired self-organizing systems, they must be able to adapt to real-world data without a teacher necessarily describing that data. In addition, supervision can sometimes force a network to adapt well to a training set but fail to produce similar results on a testing set since it is driven strongly by the training labels instead of any statistical information from the training input data which might aid in generalization.

Unsupervised machine learning is extremely valuable in the case that labeled data is not available to be learned from. In addition, unsupervised learning systems can generate unbiased internal representations by using only statistical information thus reducing the influence of the designer and increasing the influence of the data. Such learning techniques are often driven by a simple learning rule or simple metric. Also, for the purpose of showing the feasibility of self-organization, the learning mechanism must be able to create its own useful behavior from only the presented stimuli.

Many traditional unsupervised learning algorithms exist with the purpose of finding underlying structure in the presented unlabeled data. K-means attempts to partition the data into *k* groups, which is a parameter selected by the designer [92]. The Kohonen self-organizing map is a neural network which typically maps data vectors into a 2D space, contorts itself during training by updating nodes and their neighbours near the presented data, resulting in a topological map where nearby nodes respond to similar patterns [93]. Independent component analysis (ICA) attempts to separate a signal into statistically independent sub-signals which can be added to recreate the original signal [53].

Keeping with the theme of biological inspiration, its generative properties, and the robustness of the system to produce very effective results with minimal parameter selection, the restricted Boltzmann machine (RBM) and its variants are frequently chosen for unsupervised learning applications, for the reasons discussed in section 2.1.

2.3.1 Restricted Boltzmann Machine

The restricted Boltzmann machine (RBM) [94, 95] is an energy-based probabilistic network of two layers where each node is fully connected with the nodes of the other layer but has no connections within its own layer, as shown in Figure 2.10.

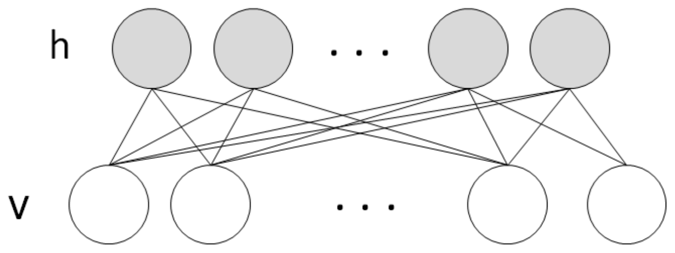


Figure 2.10. Restricted Boltzmann machine.

The RBM has two layers: the visible layer and the hidden layer. The visible layer, v, contains the nodes which are exposed to data or stimuli, and can be considered the known variables. The hidden layer, h, contains nodes which represent the feature nodes, and can be considered as the unknown latent variables. The RBM can produce hidden activations from visible inputs as well as visible reconstructions from hidden activations. The goal of the RBM is to generate vectors from the same distribution as the training data, and this goal produces useful effects. With the RBM, as the probability that the model generates the training data vectors increases, the hidden nodes come to represent statistically-prevalent features which are best suited to reconstruct that data. Those same features prove a useful representation of the training data for discriminative purposes.

In addition to producing features for discrimination, the RBM can also behave as a generative system, where it produces examples of the training dataset based on the configuration of its hidden nodes. When data is presented to the visible nodes, the network can be executed to propagate the data to the hidden layer and then propagate the hidden activations back down to the visible layer to produce new data. If the original data is inconsistent with the patterns learned by the RBM, the new data will be from the learned distribution, and can be used for applications such as denoising or the removal of outliers from data. Also, if incomplete data is presented, the complete portion is clamped and the RBM can be run to produce the missing portion. Effectively, this behaves as an autoassociative memory, where patterns can be stored in the network and retrieved by using only a fragment, or corrupted version, of the input.

The RBM is an undirected graph, meaning that data can flow both ways, from the visible nodes to the hidden nodes and vice versa through the same connections. Just like in a neural network, the weights between the visible nodes and hidden nodes measure the contribution that one node has to the other, except that in this case, the connections operate bi-directionally with the same weight.

The RBM is an energy-based model. Energy-based models consist of an energy function that produces a value for each configuration of the variables involved, where lower energy is typically associated with preferred states. Since the RBM is a probabilistic model, the energy function effectively defines the probability distribution among the variables as in Equation (2.5). When training occurs, the decrease of the energy function for preferred data corresponds to an increase in probability that the model will generate it.

$$P(v) = \sum_h P(v,h) = \sum_h \frac{e^{-E(v,h)}}{Z} \tag{2.5}$$

$$Z = \sum_{v,h} e^{-E(v,h)}$$

where Z is known as the partition function, and is the sum of the probabilities of all possible configurations. E is the energy function, v are the visible states and h are the hidden states.

In the RBM, the energy of the joint configuration of visible nodes and hidden nodes is given by Equation (2.6).

$$E(v,h) = -b'v - c'h - h'Wv \tag{2.6}$$

where v and h are the visible and hidden node states, respectively, b and c are the visible and hidden biases, respectively, W are the weights connecting the hidden and visible nodes, and the prime symbol indicates that the matrix is transposed. Ideally, the energy is lowered for preferred configurations of hidden nodes and visible nodes, while the energy is raised for undesirable configurations by modifying W, c, and b. The network encodes this energy function and tends towards configurations of lower energy when run. The goal of training is to decrease the energy of preferred configurations and increase the energy of undesirable configurations.

The network learns by attempting to reconstruct the presented data. By increasing the probability that the network can generate data samples, it will learn good features to represent that data. This can be considered from the perspective of free energy, leveraging the concept of surprise in a Bayesian simulation of the brain. To increase the probability of the training data being generated by the network, the free energy associated with that data should be reduced. This implies that the network understands enough about the data distribution that it will likely not encounter something that cannot be represented well; thus it is not surprised as a result of free energy minimization.

The formulation for the free energy of a visible vector, v, in the network, and the probability of a visible vector, is as follows:

$$F(v) = -log \sum_{h} e^{-E(v,h)} \tag{2.7}$$

By using the free energy from Equation (2.7) in the probability formulation from Equation (2.5), the following probability based on free energy is derived:

$$P(v) = \frac{e^{-F(v)}}{Z} \tag{2.8}$$

$$Z = \sum_{v} e^{-F(v)}$$

For the binary-binary RBM, where the visible nodes and hidden nodes will be binary, the free-energy formulation is as follows:

$$F(v) = -b'v - \sum_{i} \log(1 + e^{(c_i + W_i v)}) \tag{2.9}$$

Practically, to execute the RBM, the only functions required are the calculation to determine the probability that a hidden node is on given the visible state vector, shown by Equation (2.10), and the calculation to determine the probability that a visible node is on given the hidden state vector, shown by Equation (2.11).

$$P\left(h_j = 1 \left|v\right) = sigmoid(c_j + \sum_i w_{ij} v_i) \tag{2.10}$$

$$P(v_i = 1 \,|h) = sigmoid(b_i + \sum_j w_{ij} h_j) \tag{2.11}$$

where h is the hidden nodes vector, v is the visible nodes vector, and w_{ij} is the weight connecting the ith visible node, v_i, and the jth hidden node, h_j. c_j and b_i are the biases of the jth hidden node and ith visible node, respectively. In Equation (2.10), the sum is over all visible nodes, and in Equation (2.11), it is over all hidden nodes. The sigmoid function is $sigmoid(x) = \frac{1}{1+\exp(-x)}$.

2.3.1.1 Training

The goal of training an RBM is to model statistical relationships between the visible nodes by encoding them in the hidden nodes as opposed to requiring a target outcome to drive discrimination. This allows the network to determine its own representation, and is often more suitable than a user-determined representation for learning features. The goal of the training is for the probability distribution of the network on its own to resemble the probability distribution of the network when presented with the training samples.

The goal of training is to increase the probability that the network generates the training pattern, v, instead of other patterns, $(v) = \sum_h \frac{e^{-E(v,h)}}{Z}$.

The negative log-likelihood gradient, with respect to the model parameters, θ, is used to adjust the network towards producing a vector, v.

$$\frac{-\partial \log P(v)}{\partial \theta} = \frac{\partial F(v)}{\partial \theta} - \sum_{\tilde{v}} p(\tilde{v}) \frac{\partial F(\tilde{v})}{\partial \theta} \tag{2.12}$$

The two terms are considered the positive and negative phase, where v is the actual vector and $\tilde{v}$ is the reconstructed sample vector generated by the model. The positive phase increases the probability of the training data in the model by reducing the free energy. The negative phase decreases the probabilities of other samples generated by the model.

Calculating Z, or the log-likelihood gradient, $\sum_{\tilde{v}} p(\tilde{v}) \frac{\partial F(\tilde{v})}{\partial \theta}$, is intractable because it is a sum over all possible configurations of v. For practical purposes, an approximation must be used to calculate the gradient.

$$\frac{-\partial \log P(v)}{\partial \theta} \approx \frac{\partial F(v)}{\partial \theta} - \frac{1}{|\mathcal{W}|} \sum_{\tilde{v}\, \in\, \mathcal{W}} \frac{\partial F(\tilde{v})}{\partial \theta} \tag{2.13}$$

where $\mathcal{W}$ is a limited set of samples generated by the model known as negative particles. These samples are drawn from the model and provide an approximation of the model distribution to be used in place of the model's actual distribution. This is the basis of the Contrastive Divergence training algorithm.

Before discussing the Contrastive Divergence training algorithm, it is important to discuss Gibbs sampling, since it is an important mechanism for RBMs. Gibbs sampling [96] is used to sample the random variables in the network. Figure 2.11 shows an example of Gibbs sampling in the RBM.

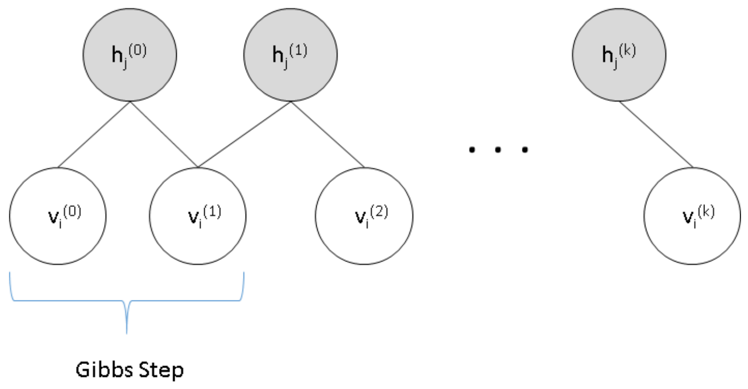


Figure 2.11. Gibbs sampling in the RBM. Example shown is between hidden node h_j and visible node v_i for k steps.

In the RBM, Gibbs sampling is performed by sampling the hidden nodes based on the visible nodes, and then vice versa. Gibbs sampling is akin to letting the network run freely as a dynamic system, where each state of the network is based on the previous state, and taking samples during this execution. Over time, if Gibbs sampling continues, the system will converge and the state of the hidden nodes and visible nodes will eventually give the actual probability distribution of the network.

The Contrastive Divergence training algorithm (CD) is used to sufficiently approximate the log-likelihood gradient from Equation (2.12) without requiring the system to reach equilibrium [95]. It implements Equation (2.13) for the RBM, where the parameters, θ, are actually *W*, *b*, and *c*.

As in Equation (2.13), CD involves calculations in two phases: the positive phase and the negative phase. The positive phase determines the response of the network when driven by the data. The negative phase determines what the network generates on its own.

CD sets the visible state to vectors from the training data and calculates the hidden state. Instead of performing Gibbs sampling until the network converges, CD performs only a limited number of steps of Gibbs sampling. Hinton shows that performing just a single step of Gibbs sampling works effectively [95].

The first sample is taken by setting the visible nodes to a vector in the dataset. The values of the hidden nodes are calculated based on the visible nodes. Here, the outer product between the visible nodes and the resulting hidden nodes is known as the positive association as shown in Equation (2.14). This association represents the correlation between visible node i and hidden node j when being driven by training data.

$$p_{ij}^{+} = v_i^{+} h_j^{+} \tag{2.14}$$

The second sample performs Gibbs sampling for one step, and is taken by reconstructing the visible nodes from the hidden nodes and then again the hidden nodes are calculated based on the new state of the visible nodes. Here, the outer product between the visible nodes and the resulting hidden nodes is known as the negative association, as shown in Equation (2.15). This association represents the correlation between visible node i and hidden node j when driven by the reconstructed data, which is representative of samples that the model generates on its own.

$$p_{ij}^{-} = v_i^{-} h_j^{-} \tag{2.15}$$

This difference between the positive and negative associations can be considered a learning signal, as the network tries to create the original data in the visible nodes from its hidden state. By adjusting the weights to reduce the difference between the positive and negative association using Equation (2.16), the system will gradually be able to reconstruct visible nodes from the hidden nodes accurately, and ultimately learn statistically significant features from the dataset. Interestingly, each weight is adjusted only by the behavior of the two nodes connecting it. Indirectly, global information is used via the nodes it connects due to their behavior of aggregating all of the connection weights, however, it is elegant in its solution that the weight only uses local information for adjustment.

$$\begin{aligned} \Delta w_{ij} &= \gamma[{p_{ij}}^{+} - {p_{ij}}^{-}] \\ \Delta b_i &= \gamma[{v_i}^{+} - {v_i}^{-}] \\ \Delta c_j &= \gamma[{h_j}^{+} - {h_j}^{-}] \end{aligned} \tag{2.16}$$

As with Equations (2.12) and (2.13), the first term increases the probability of the presented data being generated by the model while the second term decreases the probability of samples generated by the model, multiplied by a learning rate, γ. The first term decreases the energy of the preferred configuration and pulls up the energy of a nearby configuration, thus nudging the network towards modeling the desired data and away from what it currently erroneously models.

This example describes CD using one Gibbs step (CD-1), however the network can also bounce between layers a number of times corresponding to the number of Gibbs steps to be used to calculate the negative association (CD-k).

Persistent Contrastive Divergence (PCD) [97] is an extension of CD, where the negative sample is gathered after k Gibbs steps from the saved negative sample of the last update. This creates a persistent chain of Gibbs samples. For the first weight update, the negative sample is taken after a finite number of Gibbs steps starting from the training data. However, the next weight update begins the sampling with that negative sample, instead of the training data, and obtains a new negative sample after a finite number of Gibbs steps. This continues for each weight update. CD-k takes negative samples from areas in the energy function near the training data sample. The benefit to PCD is that negative samples can be taken from areas in the energy function that are further away from the training data. This has been shown to work more effectively than CD.

2.3.1.2 RBM with Gaussian Visible Nodes

The binary-binary RBM, also known as the Bernoulli-Bernoulli RBM, is useful for representing binary patterns. However, many types of data are best represented with continuous values, such as image data. The RBM can be extended to this type of data such that it contains Gaussian visible nodes while keeping binary hidden nodes. This is also named the Gaussian-Bernoulli RBM. The energy function is as follows:

$$E(v,h) = \sum_i \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_j c_j h_j - \sum_i \frac{v_i}{\sigma_i} h_j w_{ij} \tag{2.17}$$

The activation of h_j is slightly modified from Equation (2.10) to involve the variance, σ_i, as follows:

$$P(h_j = 1 \,|v) = sigmoid(c_j + \sum_i w_{ij} \frac{v_i}{\sigma_i}) \tag{2.18}$$

The sampling of v_i with Gaussian visible nodes is shown in Equation (2.19).

$$P(v_i = v|h) = \mathcal{N}(v|b_i + \sum_j w_{ij} h_j \,, \sigma_i^2) \tag{2.19}$$

where the sample of v_i is drawn from the normal distribution, through $\mathcal{N}$, with a mean of the same pre-sigmoidal activation from Equation (2.11) and a variance of σ_i.

The weight updates for CD are updated from Equation (2.16), for binary-binary RBMs, which corresponds to Equation (2.20), for Gaussian-binary RBMs.

$$\begin{aligned} \Delta w_{ij} &= \gamma[\frac{1}{\sigma_i^2} p_{ij}{}^{+} - \frac{1}{\sigma_i^2} p_{ij}{}^{-}] \\ \Delta b_i &= \gamma[\frac{1}{\sigma_i^2} v_i{}^{+} - \frac{1}{\sigma_i^2} v_i{}^{-}] \end{aligned} \tag{2.20}$$

The calculation of Δc_j remains the same as in Equation (2.16), since the behaviour of the hidden nodes remain the same. It is important to note that the learning rate, γ, must be smaller by one or two orders of magnitude compared to the binary-binary RBM for the training to remain stable. Also, learning the variances, σ, is possible with CD, but it is more practical to normalize the input data to have unit variance and manually set σ to 1.

2.3.1.3 Higher-Order RBM

Higher-order RBMs are an extension of the RBM model, and provide a powerful method of learning relationships. Instead of having the hidden nodes model statistical relationships in one set of data, the nodes will model the relationships between multiple sets of data. For example, the three-way RBM proposed by Memisevic and Hinton [61] models the joint distribution between a set of inputs, x, a set of outputs, y, and the hidden nodes, h. This three-way interaction can be thought of as a generative model which tries to predict the output, y, from the input, x, allowing the hidden nodes to develop efficient codes for the observed transformations between x and y. Keeping true to the RBM model, CD can still be applied as the training algorithm. A view of this network is shown in Figure 2.12. In the case of visual data, this can readily be applied to the concept of learning spatial transformations. The hidden nodes in this model represent the transformations between two different images.

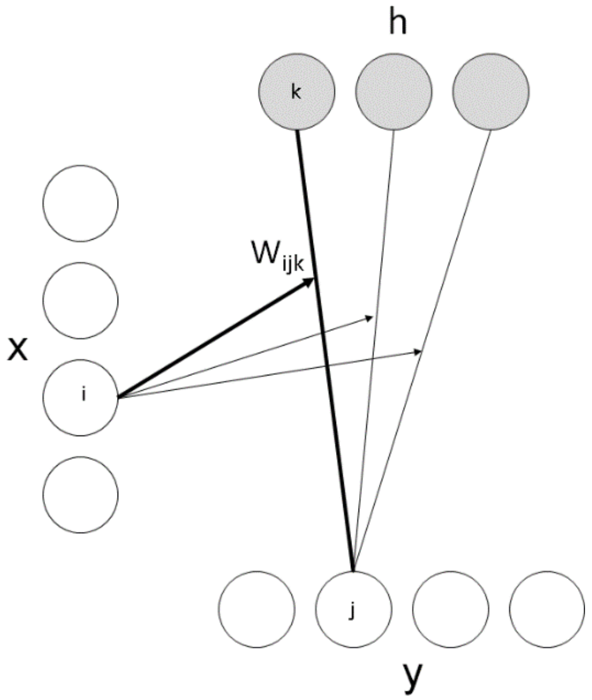


Figure 2.12. A view of the three-way RBM, where input nodes, *x*, act as a multiplier on the weights between the hidden nodes, *h*, and the output nodes, *y*. The weight, W_{ijk}, combining input node *i*, output node *j*, and hidden node *k* is highlighted.

The three-way RBM network uses three-way multiplicative interactions between nodes, specifically, pixels in the first image, x, as a multiplicative gain on the weights between the second image, y, and the hidden nodes, h. That multiplicative gain is implemented as a gate which throttles the connection between variables in y and the variables in h. The energy of the joint configuration of the nodes is defined in Equation (2.21).

$$E(x, y, h) = -\sum_{ijk} W_{ijk} x_i y_j h_k - \sum_{k} c_k h_k - \sum_{j} b_j y_j \tag{2.21}$$

where W_{ijk} are the weights connecting the input, output, and hidden nodes which are labeled x, y, and h, respectively. *c* and *b* are the biases of the hidden nodes and the output nodes respectively, since they are the only vectors that can be generated by the network.

Due to the cubic increase in parameters that occur when using larger images, a method to reduce the dimensionality of the data is proposed in the form of intermediate factors [81]. The three-way factored RBM keeps the number of parameters low with a similar efficiency in pattern learning. An example of a single factor in a three-way RBM is shown in Figure 2.13, while the energy function is defined in Equation (2.22).

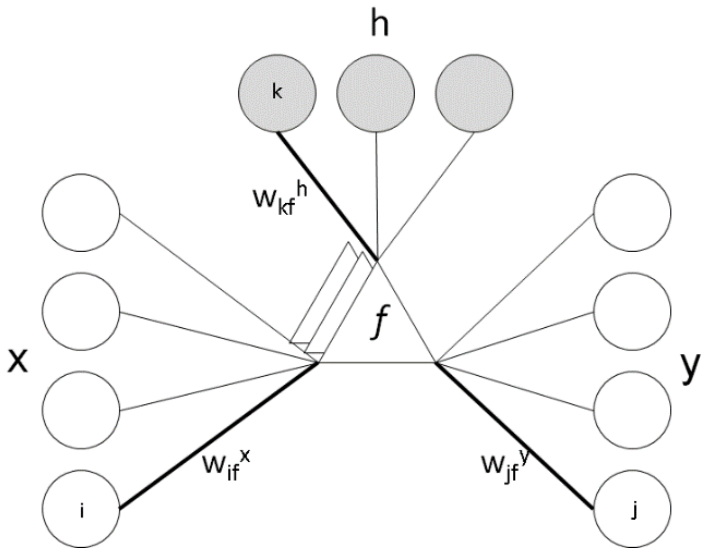


Figure 2.13. A three-way factored RBM. Around factor *f*, the weights from the ith input node to f, w_{if}^{x}, the jth output node to f, w_{jf}^{y}, and the hth hidden node to f, w_{kf}^{h}, are highlighted.

$$E(x,y,h) = -\sum_{f=1}^{F}\sum_{ijk} x_i y_j h_k w_{if}^{x} w_{jf}^{y} w_{kf}^{h} - \sum_{k} w_k^h h_k - \sum_{j} w_j^y y_j \tag{2.22}$$

Through use of the gates for multiplicative interactions and factors for reducing the number of parameters, this particular higher-order RBM is known as the Gated Factored Restricted Boltzmann Machine (GRBM). The inputs, outputs, and hidden nodes all have weighted connections with each factor. The W_{ijk} from Equation (2.21) is implemented as the sum of the factors, which compute the corresponding weight multiplication, as shown in Equation (2.23).

$$W_{ijk} = \sum_{f} w_{if}^{x} w_{jf}^{y} w_{kf}^{h} \tag{2.23}$$

To calculate an activation, each of the factors combines the required inputs through a weighted sum; combining inputs and outputs for the purpose of calculating the hiddens, or combining inputs and hiddens for the purpose of calculating the outputs. Then, each factor outputs a weighted piece of the pre-sigmoidal activation. The sum of these pieces, along with the bias, passed through a sigmoid function becomes the activation. The probability that hidden node *k* is on, given the input and output vectors, *x* and *y*, respectively, is calculated by Equation (2.24). The probability that the output node *j* is on, given the input and hidden vectors, *x* and *h*, is calculated by Equation (2.25).

$$P(h_k = 1\,|x,y) = sigmoid(c_k + \sum_{f} w_{kf}^{h} \sum_{i} x_i w_{if}^{x} \sum_{j} y_j w_{jf}^{y}) \tag{2.24}$$

$$P(y_j = 1\,|x,h) = sigmoid(b_j + \sum_{f} w_{jf}^{y} \sum_{i} x_i w_{if}^{x} \sum_{k} h_k w_{kf}^{k}) \tag{2.25}$$

The learning procedure occurs through CD, combining inputs, hiddens, and outputs to form the associations instead of just the inputs and hiddens as in Equations (2.14), (2.15), and (2.16) for the regular RBM. Just like the RBM, the updates are based on the difference between the positive association, which is driven by the data, and the negative association, when driven by the model's reconstruction. The equations for updating the weights connecting the factors to the inputs, hiddens, and outputs, is shown in Equation (2.26). The biases are updated the same way as the standard RBM.

$$\begin{aligned}\Delta w_{kf}^{h} &= \gamma[h_k^{+} \sum_i x_i^{+} w_{if}^{x} \sum_j y_j^{+} w_{jf}^{y} - h_k^{-} \sum_i x_i^{-} w_{if}^{x} \sum_j y_j^{-} w_{jf}^{y}] \\ \Delta w_{jf}^{y} &= \gamma[y_j^{+} \sum_i x_i^{+} w_{if}^{x} \sum_k h_k^{+} w_{kf}^{h} - y_j^{-} \sum_i x_i^{-} w_{if}^{x} \sum_k h_k^{-} w_{kf}^{h}] \\ \Delta w_{if}^{x} &= \gamma[x_i^{+} \sum_j y_j^{+} w_{jf}^{y} \sum_k h_k^{+} w_{kf}^{h} - x_i^{-} \sum_j y_j^{-} w_{jf}^{y} \sum_k h_k^{-} w_{kf}^{h}]\end{aligned} \tag{2.26}$$

Experimental results show that this network is able to encode transformations between pairs of images. This will be discussed further in section 2.4.6.

2.3.1.4 Sparsity

Often, it is possible that the architecture is overcomplete in the sense that the size of the representative layer exceeds the size of the inputs. This opens the possibility up that the network memorizes solutions instead of generalizing. For example, for visual recognition tasks where the hidden nodes are to behave as feature detectors, each of the hidden nodes can come to represent single pixels, contrary to the goal that feature detectors should represent common pixel relationships. Also, each node does not come to represent a certain concept or feature, and instead becomes dependent on all of the other nodes to produce a representation. To enforce that preferable relationships are detected, constraints are imposed to ensure that there are sparse activations at the feature level. Sparse activation means that only a small fraction of the nodes are encouraged to be active when presented with a certain stimulus, and nodes come to become selective and represent distinct features. This has been shown to increase discriminative power and optimize RBM representation of data [98]. This is also a principle which has been shown in biology, where neurons are very selective and only a few neurons in a group will fire for a given stimulus.

Nair and Hinton [98], as well as Lee *et al.* [77], use a technique which specifies a sparsity target, *p*, which is the desired probability of a node being active. A penalty term, or regularization term, is added during training to encourage the actual probability of the node being active, *q,* to be close to *p*. *q* is calculated after each mini-batch of training. The penalty term strength determines how aggressively the system is penalized for node probabilities exceeding *p*. It is important to choose the strength to be high enough to have an effect, but low enough to allow variability in node activations and to prevent the penalty from dominating the actual learning.

To impose sparsity, a regularization term is used as a penalty for incorrect activation levels. Essentially, the bias of each hidden node is adjusted to increase or decrease overall activation, according to Equation (2.27).

$$\Delta c_j = \gamma\left[h_j{}^{+} - h_j{}^{-}\right] + \phi(p - q) \tag{2.27}$$

where p is the target sparsity, q is the actual sparsity, and ϕ controls the aggressiveness of the sparsity term.

Goh *et al.* [99] propose a semantic distinction between the idea of sparsity and the idea of selectivity; terms which are often used interchangeably. They state that sparsity is the idea that only a few neurons are on at any given instance, while selectivity is the idea that a single neuron is only active for a fraction of the training examples. Sparsity is a property of a collection of neurons at a single instance in time, while selectivity is a property of a single neuron over a time period or multiple instances. Regularization using sparsity is alternated with selectivity to achieve nodes that are both sparse and selective by the end of training. The training is still done with CD, but instead of using the positive hidden activation, a modified activation is used as shown in Equation (2.28).

$$\Delta w_{ij} = \gamma(< v_i^+ s_j^+ > - < v_i^- h_j^- >) \tag{2.28}$$

$$r_j = (rank(h_j, \mathbf{h}))^{\left(\frac{1}{p}\right)-1}$$

$$s_j = \phi r_j + (1-\phi) h_j^+$$

The rank function assigns a value between 0 and 1 linearly based on the rank of h_j in **h**, where the lowest value gets a 0 and the highest gets 1, and p represents the target sparsity or selectivity between 0 and 1. r_j is 1 for the highest relative activation and decreases exponentially towards 0 as the activation rankings get lower, with a falloff proportional to $1/p$. s_j is the modified activation, which combines the sparsity/selectivity regularization with the actual positive activation by some mixing parameter, ϕ. If ϕ is 0, no regularization is used, and if ϕ is 1, the regularization is used instead of the original hidden activation; this controls the impact of regularization, similar to the aggressiveness from Equation (2.27). Finally, the weight update rule continues normally, except the modified activation is used instead of the positive hidden activation.

2.3.1.5 Biological Plausibility

The relationship of the free-energy principle to RBMs is defined in several places by the free-energy equation governing the RBMs visible and hidden nodes. The idea that the system behaves generatively to both produce data similar to what it has been trained on and learn features that represent data is an important concept for unsupervised learning and the application of a Bayesian model to the biological brain. The RBM attempts to represent training data such that new data is not surprising to it, and that it can represent any data from the same distribution. In some sense, this is a measure of understanding.

The idea of learning via the adjustment of connections between neurons, based on the Hebbian rule, is consistent with the weight modification using only the connected nodes in the RBM. It is more biologically-plausible that error signals stay local within a connection than the idea that they are propagated through a longer path involving many neurons and synapses as they would be if they followed the backpropagation neural network concept.

Unsupervised learning is also a biologically-plausible concept, especially from the perspective of self-organizing networks in the brain. Also, it is sensible, since it is unlikely that there is an explicit supervisory signal which drives learning of features. In addition, the probabilistic nature of the RBM is consistent with the idea of noise and inherent inaccuracies in real-world systems, and is used such that it is actually beneficial to produce better features.

Conceptually, the discussed RBM techniques have biological parallels and follow the free-energy theory based on their generative properties and their training which acts to reduce surprising stimuli.

2.3.2 Deep Learning

One of the benefits of the RBM is that it is able to learn relevant features as it statistically models relationships directly from the data. It is then possible to layer several RBMs on top of each other, where the hidden nodes of one layer behave as the visible nodes of the next, as shown in Figure 2.14.

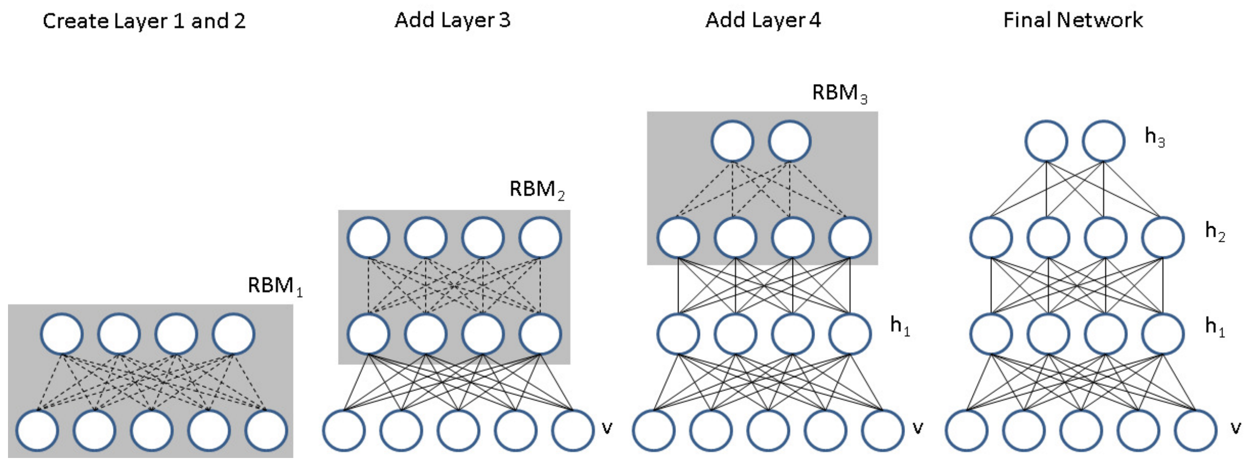


Figure 2.14. Construction of a deep architecture (5-4-4-2 nodes per layer) using RBMs. Each RBM is trained using the outputs of the layer before it, where dotted connections are being trained and solid connections have already been trained. v is the visible layer, and h_1, h_2, and h_3 are the hidden layers, with h_3 being the network's final representation of the data. The complete network is shown on the right.

Each RBM is trained in a greedy manner, meaning that it is not concerned with learning global structure or getting global error signals, and only learns based on data from the layer below it, its own hidden layer, and its own weights. The hidden nodes of the top layer form the network's final representation of the source data. This type of layering allows the network to learn its own statistically-relevant features at each level, based on the statistically-relevant features from the layer below, and so on. Ultimately, the network encodes information about the data without any supervision and without any predetermined bias of what features might be relevant. This type of architecture is considered a deep architecture, where the network extracts its own complex hierarchical representation of the data.

When the network is converted into a feedforward neural network, using the same weights, it is considered a Deep Neural Network (DNN). This is one of the most popular methods by which stacked architectures have been used, and literature shows a notable improvement in accuracy and training time over traditional neural networks. Bengio *et al.* [100] show that this type of layer-by-layer training can actually help initialize the network to a better state such that backpropagation can take over and achieve better results than networks that use backpropagation without the pre-training. The issue that backpropagation could not effectively train deep networks is mitigated by this technique and is shown to reduce error rates significantly. Hinton *et al.* [21] use this pre-training to decrease speech recognition error rates by a relative amount of 23%.

2.3.2.1 Generative Model

Stacked RBMs can also be used as a generative model, where the network produces examples of the data it is trying to model. By presenting input data, activations propagate upwards until the top most layer, at which point the activations can propagate downwards until they generate examples at the input layer. This type of network is known as a Deep Belief Network (DBN). They are best described as multi-layered networks which can be used for classification as well as generation. DBNs have their top two layers form an undirected symmetric graph representing an autoassociative memory, while the layers beneath them are successively activated by top-down connections from the layer above, as shown in Figure 2.15.

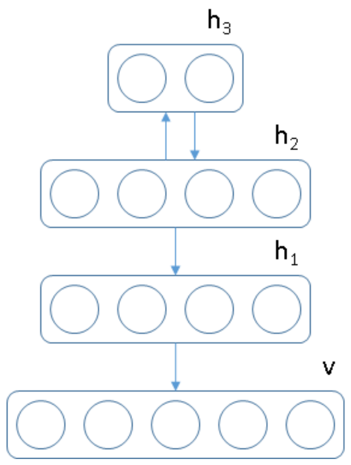


Figure 2.15. Deep Belief Network.

Given a state in the top-most layer, representing a set of features, the DBN will generate activations in a top-down manner until it reaches the bottom-most layer. The result of the bottom-most layer is the generation of a possible input vector corresponding to the feature set at the top-most layer. Hinton [101] uses a generative network to classify hand-written digits from the MNIST dataset [102] as well as generate samples of them. This model uses 4 layers, with the input layer taking in the image, and the three subsequent hidden layers forming the internal representation. There is also a 10-visible node extension which connects to the top-most layer, which represents the classification labels associated with the examples. This architecture is shown in Figure 2.16.

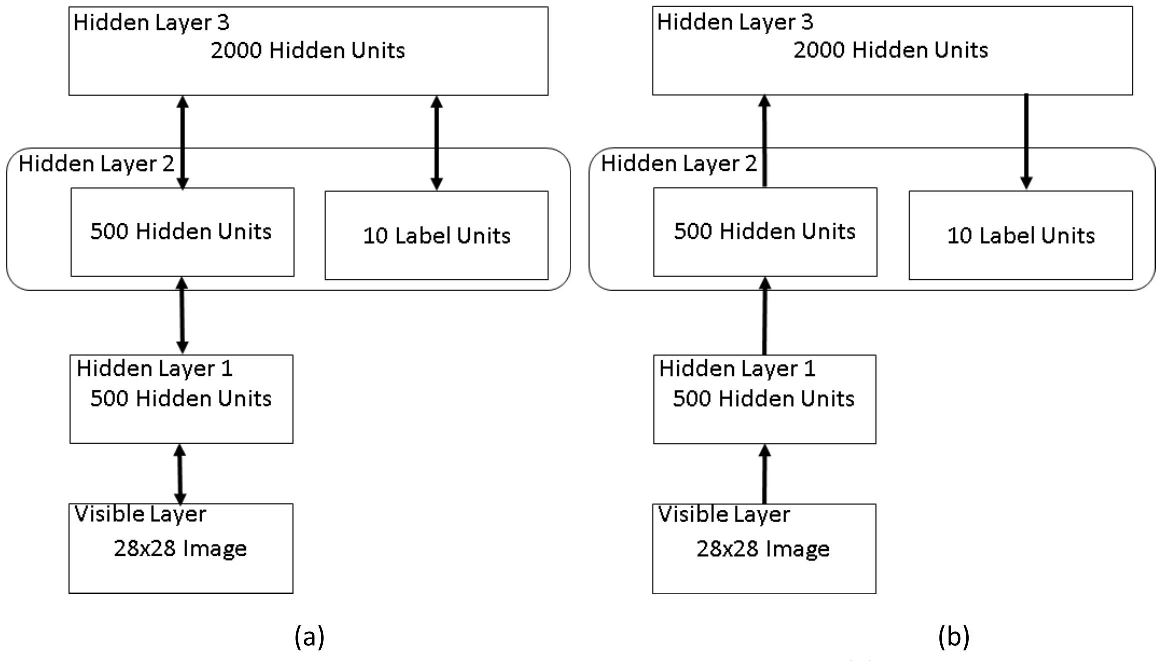


Figure 2.16. Deep Belief Network for associating handwritten digits to labels: (a) actual structure of the network for training, with bidirectional connections, and (b) same network used for classification, with the label nodes being generated.

To train the network, as before, each RBM is trained using the data below it. For the top-most RBM, during training, the network concatenates the 2nd hidden layer output with the 10-visible node label extension, to form the visible layer. Each of the 10 nodes in the label layer corresponds to a digit class. During training, when an image is presented at the input layer, the appropriate class label is also activated while the other labels are set to 0. This allows the top-most layer to encode the relationship from the 2nd hidden layer features and the appropriate labels as a bi-directional autoassociative memory. For classification, when the examples are presented at the input layer, the activations are propagated upwards through the network until it reaches the top-most layer. With the 2nd hidden layer outputs clamped, the top-most RBM is allowed to run and generate its unclamped visible nodes. This generation activates the label node which corresponds to the input data's classification. This same principle is used in the motion tracking model proposed in Chapter 4, using concatenated bidirectional visible nodes to both learn patterns and create them.

2.3.3 Reinforcement Learning

Reinforcement learning is a term used for both a biological principle as well as the machine learning models that mimic this principle. The idea behind reinforcement learning is that a model attempts to take actions that maximize its reward [24], which it learns to do based on trial-and-error interactions with its environment. The model is exposed to some stimuli and produces results, or actions in this context. It then learns from those actions based on a given reward or objective, such that it eventually produces the desired action given the stimuli. The reward is typically either positive when the model does what it should or negative when the model does what it shouldn't. Though this might seem like a trivial learning task, similar to a coarse error function used for backpropagation, there are some key differences that make this more challenging. Often, the system is temporal, meaning that the stimulus does not directly produce the action, but is a part of a longer chain of causality, including things such as previous states, previous stimuli, and previous actions. Also, the reward is often not known at the time of the stimulus, so it requires application of this reward through time to the previous states of the system. Also, since the reward is not explicit in identifying which parts of the causal chain were important for the reward, it requires some disassociation to determine which stimulus-action associations require strengthening and which require weakening. Lying somewhere between supervised and unsupervised learning, reinforcement learning succeeds in situations where labeled data is scarce or inaccurate, and is concerned with online learning in an environment where it can act on its environment and receive feedback.

Reinforcement learning has recently been paired with deep networks to produce groundbreaking results in the domain of game playing, where human-level performance has been achieved or surpassed [103, 104]. The novelty is that actions are learned from raw data, such as a board state in a board game or video frames of a video game. The deep Q network applies a deep network to estimate what the future reward of each action is based on the previous state and the current state, which learns online to eventually produce actions corresponding to the greatest future reward.

2.3.4 Summary

RBMs were discussed as an efficient method to reduce dimensionality and form a compact representation of data for classification. The probabilistic unsupervised network is capable of learning statistically-important features from unlabeled data, producing biologically-consistent results when applied to real-world data. RBMs have also been used to create sparse higher dimensional representations to better highlight importance in the data. The RBM can learn its own features and represent the input data differently, often in a statistically more relevant way. However, depending on the type and quantity of data, it may be impractical or impossible for an RBM to represent complex relationships in the data, such as hierarchical relationships where important features are composed of other lower-level features. The stacked RBM addresses this issue and forms the basis of deep architectures, where greedily-trained individual RBM layers can be stacked on top of each other, and where the output of one layer becomes the input to the next layer. This stacked architecture can represent complex relationships at increasing levels in the hierarchy.

Another benefit is that the RBM, and ultimately the stacked RBM, can behave generatively by reproducing data similar to the training dataset. It can even remove noise and corrupted areas to reconstruct the input based on what it understands about the data distribution as an autoassociative memory. It has even been shown that it can use its generative properties for classification without explicit supervision by concatenating labels to the input vector during training and reconstructing those inputs during testing. The ability to act generatively is very important to the novel work in this thesis.

The ability to extend the RBM paradigm to using real-valued inputs through Gaussian visible nodes makes it an excellent method to represent visual data. The extension to factor three-way interactions allows the network to realize higher-order concepts, such as the learning of transformations between two sets of images without learning about the images themselves.

Reinforcement learning was also reviewed, which is a method driven by a reward which forces itself to readjust its behaviour to maximize that reward. It is biologically rooted, similar to how the brain seeks out actions that allow it to receive a reward, and avoids actions that produce a punishment. In the context of this thesis, the reinforcement learning principle is valuable, in that we will also be using a coarse indirect learning signal which can be considered a reward. That reward will modulate the learning such that the model learns to prefer certain actions over others given a certain stimulus; when there is motion, it prefers to move its gaze such that the object stays the same inside its receptive field. The successful prediction of movement is correlated to the projection of the object in the receptive field, the similarity between those projections over subsequent timesteps and gaze control actions is considered the reward. The differences are that the model presented in this thesis operates over one timestep, so it is a simpler problem and does not require a state history. Also, it produces actions through the same nodes that it receives stimuli, resulting in a bi-directional model as opposed to a feedforward model with recurrent components.

2.4 Unsupervised Feature Learning

Machine vision has historically used handcrafted feature detectors and custom classification algorithms to accomplish recognition tasks. Over the past decade, with the increase in computational power, availability of real-world data, and improvements in machine learning, the movement to learn features from real-world data has yielded more discriminative and more robust features than ever before. Without a reliance on custom architectures and designer-influenced feature selection, unsupervised learning algorithms can learn valuable features directly from the data.

A number of techniques based on Hubel and Wiesel's hierarchical concept of early vision [55] have gained traction over the past three decades. The idea of simple cells and complex cells forming a hierarchy, achieving robustness through invariance and memory through the adjustment of connections, is a computationally-feasible approach since it can extend the connectionist paradigm of artificial neural networks. Without going too in-depth into the plethora of multi-layered architectures and parameter tweaks that produce increasingly better classification results, this section will focus on the basic algorithms which are the building blocks of these architectures and the ones that are actually used in this thesis to learn the best visual features from the data in an unsupervised manner. In addition to visual features for the purposes of classification, learning from videos to encode image transformations will also be investigated.

This section will also discuss pooling, which is a method of combining local information in a neighbourhood; typically either the neighbourhood is spatial, temporal, or in feature space. Pooling functions combine several responses to produce a single value with the goal of removing unnecessary information and preserving important information. Pooling is often used to add stability, improve invariance, or reduce dimensionality.

The visualization of features in this thesis, such as the example shown in Figure 2.17, can be interpreted as displaying how the corresponding feature detector, or filter, calculates its response. Brighter areas stimulate the response, darker areas inhibit the response, and gray areas are of no effect. The degree of brightness or darkness indicates the degree of stimulation or inhibition, respectively, that a given area has on the feature detector's response. This visualization also indicates what the optimal stimulus of that feature detector is; a feature that best matches the stimulating area can be considered the optimal stimulus. Though there are many possible optimal stimuli, considering that some visual areas have no effect on the feature detector's response and they can be of any value, the described concept of the optimal stimulus will be used in this thesis. Technically, filters exist to detect specific image features;

filters can also be considered feature detectors. In this thesis, the terms filters, features, and feature detectors will be used interchangeably.

2.4.1 Sparse RBM and Stacked RBMs

Considering real-world image datasets, Lee *et al.* [77] use a sparse RBM to develop feature nodes that respond to features similar to Gabor filters. An implementation of this approach developed in the scope of this thesis led to the results shown in Figure 2.17. This efficient coding is a better representation of the data than the non-sparse RBMs, where hidden nodes represent less visually-relevant patterns.

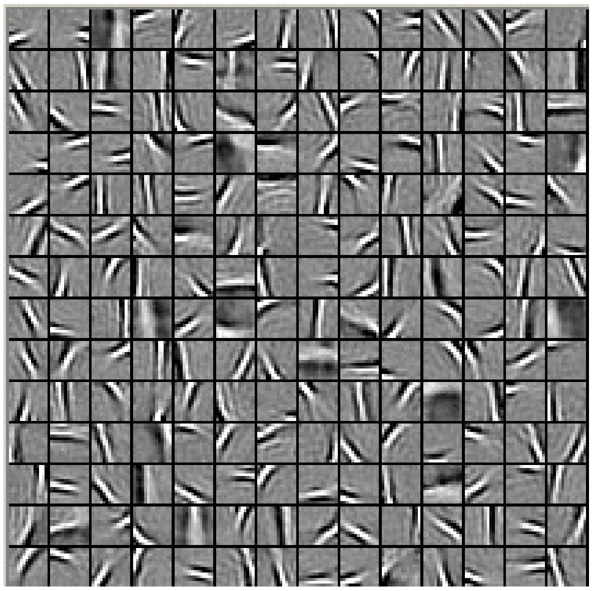


Figure 2.17. A 14x14 array of Gabor-like features learned by sparse RBM on the van Hateren natural image dataset [105] (stimulation in bright areas, and inhibition in dark areas).

Lee *et al.* stacked another sparse RBM on top of that sparse RBM to learn more complex features from real-world image data. The second RBM learns more complex features based on the first RBM's features when exposed to the same image data, such as corners and junctions. This shows that each layer can integrate information from the previous layer to craft more complex representations at increasing levels.

2.4.2 Independent Component Analysis

Despite the lack of biological plausibility in their processes, it is important to discuss a set of signal processing techniques that have had success in extracting results from visual data consistent with results in physiological experiments. Independent Component Analysis (ICA) is one such unsupervised learning technique that has shown promise. This statistical technique is used to separate a dataset into independent subcomponents, considered features, as if the original dataset is comprised of a linear mixture of said subcomponents [53], in the form of Equation (2.29).

$$\begin{gathered} x = As \\ s = A^{-1}x \end{gathered} \tag{2.29}$$

where x is the observed dataset, s is the estimated subcomponents, and A is the mixing matrix which mixes the subcomponents linearly. s and A must both be estimated. Centering and whitening are both required as preprocessing steps for ICA.

ICA then rotates the data such that the Gaussianity of its projection is minimized in all axes. This rotation determines the independent components. Assuming that real-world image patches can be composed of linear combinations of different features, ICA is a good mechanism for determining what those different features are. In experiments with real-world scenes, the independent components tend to resemble Gabor filters of different orientation, position, phase, and scales [106]. An implementation of this approach realized in the scope of this thesis produced the results shown in Figure 2.18.

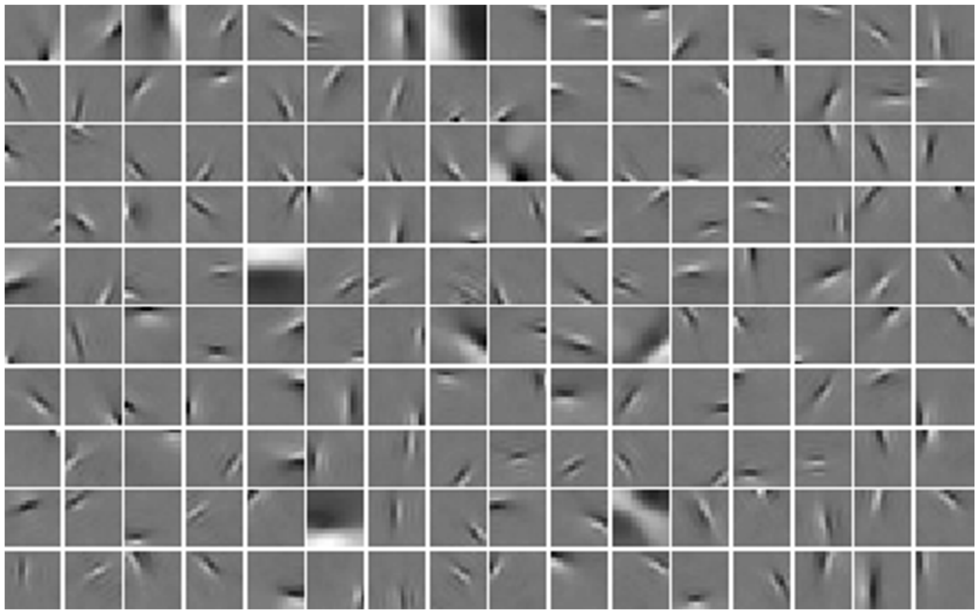


Figure 2.18. Filters learned by ICA from real-world images (stimulation in bright areas, and inhibition in dark areas).

2.4.3 Learning Invariance

Visual invariance is often credited for improving the effectiveness of feature extraction models for classification purposes. For example, the same object in different positions, orientations, or scales should provide the same signature from the perspective of visual feature detectors such that an object can be classified accurately despite its pose. In the real-world, objects undergo many different transformations, yet the biological visual system is still robustly able to identify them. A visual classification model should have the same property. In addition to boosting accuracy, a feature that is selective to an important visual characteristic but invariant to unimportant ones is also efficient in that fewer detectors are required to describe an object. For example, instead of having a feature detector for an object at one position and a different feature detector for that same object undergoing a slight transformation, having one feature detector which responds strongly to the object under both configurations is a much more desirable property. This thesis investigates this concept in Chapter 3, and attempts to decipher how much of an effect invariance truly has on classification, and how to learn features that balance selectivity and invariance directly from real-world visual data.

The most effective use of visual invariance has been through the popular Convolutional Neural Network (CNN). LeCun and Bengio [107] proposed the CNN for pattern recognition which implements an architecture inspired by the Hubel-Wiesel hierarchical framework through a hierarchical feedforward neural network trained by backpropagation. It uses alternating layers of simple cells and complex cells. Theorizing that alternating layers of simple cells and layers of complex cells compose the visual hierarchy, based on the idea that hypercomplex cells in the visual cortex have a similar relationship to simple and complex cells [108-110], such a design is adopted for the computational model.

The CNN consists of alternating simple cell and complex cell layers, where each layer contains multiple planes. Simple cells in the CNN are modeled as feature nodes, and take inputs from a part of the overall image, considered the receptive field. Biological simple cells are tuned to respond to very specific features in the input image, providing simple feature extraction for patterns such as edges, line orientation, and corners. In the CNN, the feature node is generalized. It will respond to specific features in the input image at the first layer, and at successive layers it will respond to a specific feature from the previous layer. Each plane in the CNN serves to represent a specific feature, with each node in the plane looking for that feature at a different location in the previous layer. Complex cells in the CNN are modeled as feature pooling nodes, which respond to the outputs of features nodes that detect the same feature at spatially adjacent positions from the corresponding plane in the previous layer. Each pooling node provides translational invariance to features within its receptive field. All pooling nodes in the same plane have static weights, except for the trainable bias, hence another reduction in parameters for efficient use of the backpropagation.

Unlike previous attempts by backpropagation feedforward neural networks for visual classification tasks, this network uses weight sharing in addition to its biologically-inspired complex cells and hierarchical structure. In addition to its elegance in performing translation invariance, weight sharing is particularly efficient here because of its usefulness for the acceleration of backpropagation learning due to a reduced number of parameters, resulting in a significant decrease in training time. Weight sharing provides a simple way to know which features must be grouped together to create invariance, and the hard-wiring to the pooling nodes actually creates that invariance. There is no evidence to suggest that weight sharing is biologically plausible, and the hard-wiring limits the model to certain assumptions and user-designed parameters; with the primary assumption being that invariance is best suited to only translation invariance at each level of the hierarchy. The biggest benefit of the recent trend in deep learning is the ability to learn relevant information directly from the data without supervision, and weight sharing is a hindrance to that concept. Complex cells are often resistant to variations other than just translation, and the below-mentioned works shows that other types of invariances can be learned also.

When using static images to learn invariance, the idea of overlapping information or dependence is used as supplementary information. Kavukcuoglu *et al.* [111] induce 2D topography into predictive sparse decomposition (PSD) to further learn invariant features. By promoting dependence in neighbouring features, yet sparsity in separate neighbourhoods, the network generates a 2D topography where features smoothly vary across the hidden layer. The standard PSD cost function is updated with the neighbourhood dependence results in Equation (2.30), where the second term induces the topography.

$$J = \|x - Dz\|_2^2 + \lambda \sum_{i=1}^{K} \sqrt{\sum_{j \in P_i} N_j z_j^2} \tag{2.30}$$

where x is the input vector, D is the dictionary to be learned, z is the activation vector, K is the number of patterns and N are the fixed neighbourhood weights which controls the impact of the surroundings on each activation. Similarly, Goh *et al.* [112] introduce a 2D topography into an RBM, which learns invariant colour features that fluctuate smoothly over the hidden layer. It is also claimed that inducing sparsity and topography produces more invariance in features than without.

Independent Subspace Analysis (ISA) is an extension of ICA which assumes the subspaces are independent, yet finds components inside each subspace which are not necessarily independent [113]. Each subspace contains several related components, often providing information on directions of invariance. The relation of the components in a subspace is determined by higher-order correlation. As a result, components that are typically active simultaneously are grouped together to form a subspace. The importance of this method is that these subspaces can represent complex cells in the visual system, showing that pooling nodes for the purpose of invariance can indeed be learned. ISA has been used on static images as well as video [114] to learn features and invariances through pooling. Examples of these subspaces extracted from real-world images from an implementation of ISA developed in the scope of this thesis are shown in Figure 2.19. It can be seen that each feature within a space is slightly different from each other feature in the same space, with variations in position, orientation, phase, and/or scale.

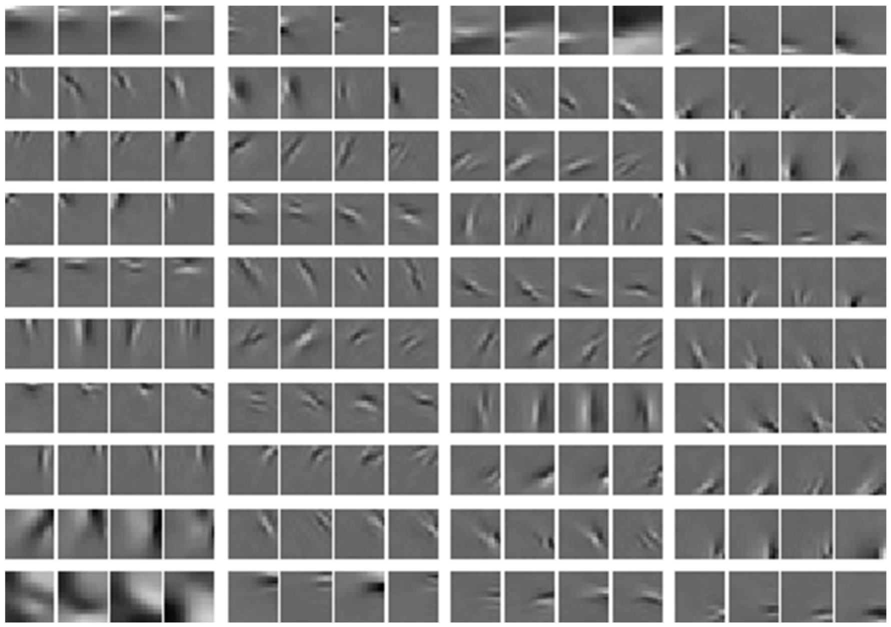


Figure 2.19. Subspaces learned by ISA from real-world images. Each group of 4 contains the related filters which compose a subspace (stimulation in bright areas, and inhibition in dark areas).

Topographic independent component analysis (TICA) is also an effective tool to determine subspaces, but its elegant formulation allows a topographic relationship between all of the components, where components outside of a neighbourhood are independent, but components within a neighbourhood are not [79]. It provides much of the same information as ISA, but expands it to the idea of producing a 2D topological structure of simple cells similar to the topological structure found in V1 with the additional idea that any local grouping can behave as a complex cell where the filters are invariant to a small degree of transformation. An example of TICA results obtained from our implementation, also trained on real-world images which smoothly vary to cover a wide variety of orientations and positions, is shown in Figure 2.20.

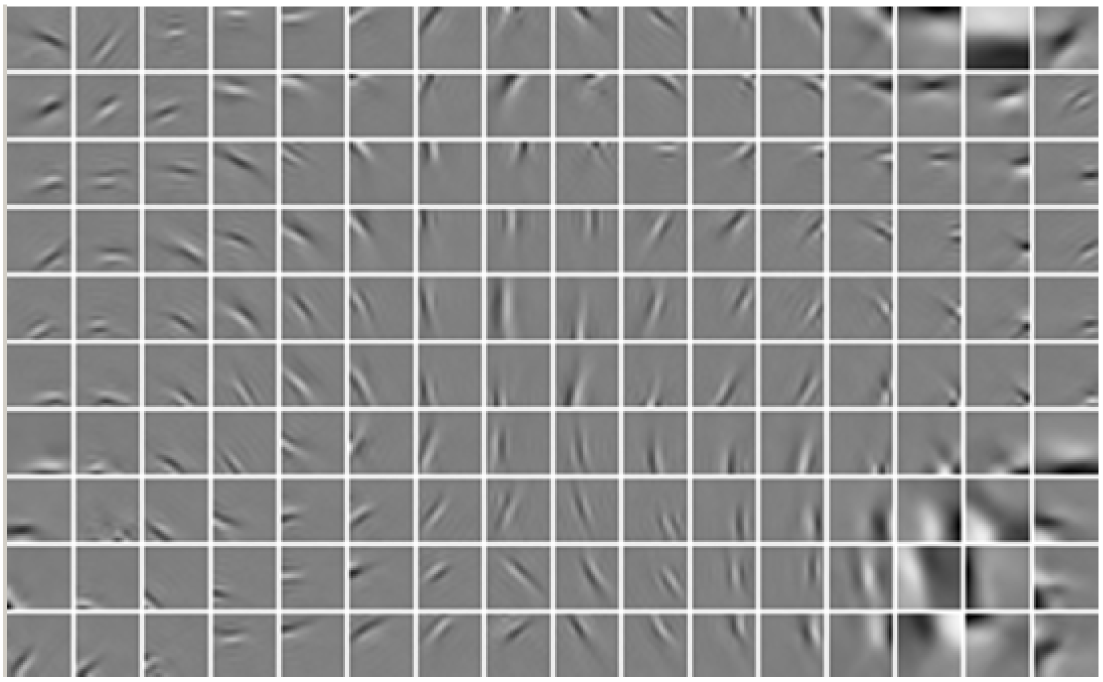


Figure 2.20. Topological map produced by the TICA self-organizing algorithm, with nearby nodes preferring similar orientations and positions (stimulation in bright areas, and inhibition in dark areas).

To show invariance properties of certain nodes, they can be graphed with respect to the transformations of optimal stimuli. These transformations are often based on phase, position, and orientation. Peak response of the filter should occur with the optimal stimulus, and the lesser the degradation of response due to varying parameters, the larger the filter's invariance is. Naturally, with greater invariance, the feature becomes less selective, will respond to more features, and thus might be a hindrance for discriminative purposes. Therefore, some balance between selectivity and invariance must be maintained for discrimination. Theoretical responses (y-axis) of a linear filter and a translation-invariant filter given translations and rotations of the optimal stimulus (x-axis) is shown in Figure 2.21. The examples show that the translation-invariant filter degrades much more slowly as the position is varied, while the linear filter degrades very quickly as the optimal stimulus becomes translated. However, for orientation, they both remain equally selective, as they both degrade when the orientation varies slightly. This is the type of behaviour that is desired, where a feature detector will be sensitive to one type of transformation of a feature, but robust to another.

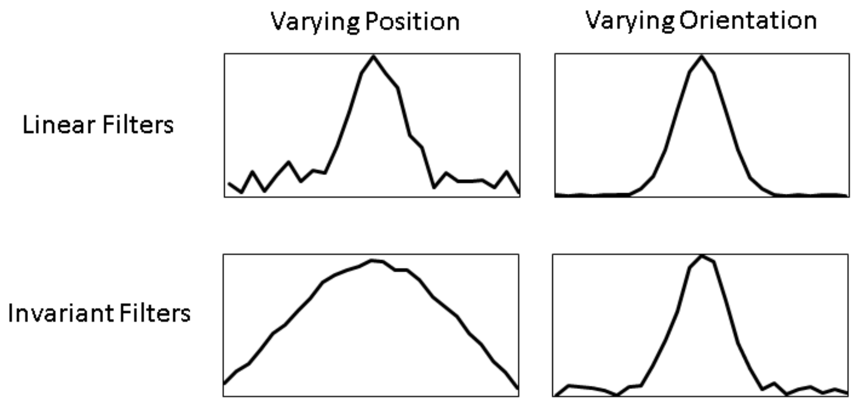


Figure 2.21. Example responses of: linear filters, representing simple cells (top) and invariant filters, representing complex cells (bottom), when exposed to optimal stimuli under varying position and varying orientation. These examples are based on ISA linear filters and invariant pooled ISA subspace filters, but the complex cell patterns should emerge in any method to generate invariant features, such as pooled TICA neighbourhoods.

ISA and TICA have shown results that are physiologically-consistent with their ability to develop and group common features such that invariance can be achieved. Implementing unsupervised learning in the form of TICA and pooling features in a neighbourhood to achieve invariance, the large scale network crafted by Le *et al.* [22] show that a more biologically-plausible network, with more invariance than just translation invariance, can be a very effective system. Le *et al.* propose a massive deep belief architecture to learn representations from real-world images. Arguing that weight sharing is less biologically plausible, the network does not use convolutions, but uses learned invariances in its place for pooling. This way, different types of invariances can be learned beyond just the standard translation invariance of convolutional models. Using large receptive fields, sparsity constraints, contrast normalization, and multiple stacked layers, they produce one of the largest vision-based networks, containing over 1 billion trainable parameters. As with all deep belief networks, training is done layer by layer. TICA is used to learn the feature nodes and neighbouring features are pooled such that they can provide invariance in an unsupervised manner and reduce the dimensionality of the data for subsequent layers.

Using the large ImageNet datasets [115], which contains approximately 14 million images in 22000 categories, the network described above, by Le *et al.* [22], achieves 15.8% classification accuracy. The state-of-the-art prior to this of 9.3% was produced by Weston *et al.* [116], meaning this new network produces a 70% relative improvement. It is important to note that unsupervised pre-training was very important to achieving these results. The same network with random weights before fine-tuning was able to only achieve 13.6% accuracy. Applied towards face detection, images from the Labeled Faces in The Wild dataset is used [117], which contains non-aligned human faces in real world images. The network was to determine whether a face was in the image or not. A random guess is accurate about 64.8% of the time. Without any supervised training, the best neuron in the network was able to detect faces with 81.7% accuracy. The best linear filter was found to be 74% accurate. This network was also applied as a cat detector, using cat faces from Zhang *et al.* [118] dataset, and negative samples from the ImageNet dataset. The network obtains 74.8% accuracy compared to the best linear classifier's accuracy of 67.2%. When applied as a human body detector, from a benchmark dataset [119], the network obtains 76.7% accuracy compared to the best linear classifier's accuracy of 68.1%.

These results show the power of unsupervised layer-wise pretraining, and the tremendous increase in efficiency when applying it to a large-scale network. It also shows that weight sharing is not necessary to achieve good deformation-resistance, and that learning invariances in an unsupervised manner can produce very effective results.

There is also a belief that complex cell wiring can be learned by evaluating slow changing features in real-world video. The response of individual photoreceptors or pixels change quite frequently over time as they are sensitive to small changes in the environment. However, actual objects, and their internal representations in the brain, are much less sensitive to immediate changes in the environment and often appear, move, and disappear relatively slowly. The natural extension is that certain features will change slowly over time in real-world videos, and these gradual changes can be captured and used for invariance, not just temporally, but in a static image also.

Földiák [120] was among the first to use a temporal consistency component for learning invariance in a network from sequences of data, with winner-take-all competitive learning. Without weight sharing, the network is allowed to learn features for its first layer as normal, and the second layer, which represents complex cells, groups similar features together based on a learning principle coupled with the patterns found in real-world motion. Földiák describes that instantaneous Hebbian learning for the complex cell would not be effective, since it would only measure overlapping patterns and not necessarily changes in patterns over time. In order to track activation over time, Földiák introduces a trace variable, $\bar{y}$, which takes a running average of the activation, y, subject to Equation (2.31).

$$\bar{y}^{(t)} = (1-\delta)\bar{y}^{(t-1)} + \delta y^{(t)} \tag{2.31}$$

where δ measures the influence of the current activation on the running average, and y is the current activation of the node. This method of low pass filtering will suppress fast changing features while preserving slow changing features. For Hebbian learning, the trace variable is then used instead of the regular activation for the competitive learning.

The method is not applied on real-world video sequences but rather on fabricated sequences involving a line sweeping across the receptive field. Despite the unrealistic training scenario, the results in this early paper show that learning of invariances by capturing slow changing features is possible.

Einhäuser *et al.* [121] attempt to acquire similar measurements from their computational model as the ones found in Hubel and Wiesel's original findings on the cat visual cortex by using real-world video captured by mounting a camera on a cat. Their approach is also similar to that of Földiák's, but they introduce a larger scale model and apply it to real-world video with online learning, using only one frame of video before each update. Their first layer behaves as simple cells, learning oriented, positioned filters, while the second layer behaves as a layer of complex cells which are fully connected to the first layer. Their results show that the second layer, using the adapted trace variable for temporal smoothing, connects to similarly oriented features. It was found that orientation of features change much more slowly than their position, and as a result, the second-layer nodes are tuned to nodes of similar orientation but varying position. This paper shows that it is possible to learn the behavior of complex cells using just the temporal structure of real-world video.

Zou *et al.* [122] use temporal slowness in conjunction with a deep belief network to show that unsupervised training can learn increasing invariances as the network aggregates more and more layers. Building upon the autoencoder [123], the cost function is regularized with an L1 temporal difference cost as shown in Equation (2.32).

$$J = \sum_{t=1}^{N} \left\| x^{(t)} - W^T W x^{(t)} \right\|_2^2 + \lambda \sum_{t=1}^{N} \left\| p^{(t)} \right\|_1 + \gamma \sum_{t=1}^{N-1} \left\| p^{(t)} - p^{(t-1)} \right\|_1 \tag{2.32}$$

where the first summation is the typical autoencoder, the second summation uses the L1 norm to induce sparsity, and the third summation is the temporal coherence term, which emphasizes slow changes in activations. W is the weight matrix, $x^{(t)}$ are the input features at time t, $p^{(t)}$ are the pooled hidden activations at time t, which will be discussed further in section 2.4.4, by $p^{(t)} = \sqrt{H(Wx^{(t)})^2}$ where H selects the neighbouring features which p should pool. With a non-overlapping neighbourhood size of two, the network learns pairs of features which their respective pooling node becomes invariant to. This joint learning of features and invariance is different from the layer-by-layer alternation of detection and pooling used by previous methods.

Saliency-based feature detection is used to identify areas of interest in the van Hateren video dataset [124] and motion tracking is used follow that area over time, thus exposing the network to only slow-changing features. The first set of feature and pooling layers, trained using temporal slowness on the van Hateren natural image dataset [105], learns oriented features like other unsupervised learning techniques applied to real-world image data, with the exception that they are more translation invariant. The second set of feature and pooling layers is trained on the outputs of the previous layer, after principal components analysis (PCA) to reduce the dimensionality, in the same manner. The training of the second layer gives rise to the further learning of different invariances beyond translation, including rotation, warping, and larger translations. This shows that more complex invariances can be learned using temporal slowness at deeper levels of the network.

When applied to static image datasets, there is a notable increase in classification accuracy when using these learned invariances as opposed to hard-wired translation-only invariance. Classification is done by concatenating activations from the first layer and second layer, and pooled over an unspecified size of spatial regions. On the STL-10 [23], COIL-100 [125], and Caltech 101 [126] datasets, an increase of 4-5% in classification accuracy was found when training with video over training with static images. Note that the training video is from a different dataset to the test images, which means that the features learned from real-world video are sufficiently similar to work with an unrelated dataset.

The idea of learning invariance from either images or video will be explored in this thesis, but from the perspective of biasing a standard neural network model enough that it learns invariance from exposure to real-world data.

2.4.4 Pooling Functions

Pooling is an important concept to reduce the dimensionality of data, which simultaneously provides a more efficient informative representation as well as potentially incorporating spatial information to improve representation such as tolerating small transformations and resulting in invariance. In the CNN, though the behaviour of feature nodes resembles a typical artificial neuron, the behaviour of pooling nodes requires further investigation. The original CNN modeled the pooling node like a standard artificial neuron, with its output being a function of the sum of its weighted inputs, where the weights are all equal and normalized as formalized in Equation (2.33).

$$h_{i+1,x,y} = sigmoid(\frac{1}{n}\sum\nolimits_{(j,k)\epsilon\, N(x,y)} h_{i,j,k}) \tag{2.33}$$

where $h_{i+1,x,y}$ is the value of the pooling node at layer i+1 and position (x,y), *n* is the number of values in the neighbourhood, and the sigmoid function takes the sum of *h* over all coordinates (j,k) within the neighbourhood of (x,y). With this method, the pooling node averages the weighted inputs before providing them to the sigmoidal function. The benefit of this pooling mechanism is that all inputs are treated equally, and if they produce strong outputs, the pooling node will also produce a strong output. Averaging without the sigmoidal function, shown in Equation (2.34), has also been investigated [127].

$$h_{i+1,x,y} = \frac{1}{n}\sum\nolimits_{(j,k)\epsilon\, N(x,y)} h_{i,j,k} \tag{2.34}$$

Coates *et al.* [23] make use of a simpler pooling method, where the full image is divided into regions, and the sum of the responses within those regions is used. Features are extracted from a window of a specific size, w, and shifting with a stride, s. Thus, an image of n-by-n pixels with d channels produces an ((n-w)/s+1)-by-((n-w)/s+1)-by-d representation with K feature planes, where each feature plane represents a single feature detector's activations at each location of the convolution, similar to feature planes in CNNs. The channels are usually red, green, and blue for colour images. Formally, y_{ij} is used as the K-dimensional representation extracted from position (i,j) in the image. A simple pooling mechanism is implemented to reduce the dimensionality of the representation by summing over local regions. Specifically, the image is divided into 4 equally-sized quadrants, and the feature vector is reduced from a ((n-w)/s+1)-by-((n-w)/s+1)-by-K representation to a 2-by-2-by-K representation. The sum of the y_{ij}''s in each quadrant makes up the final 4K representation. This process is shown in Figure 2.22.

It is important to note that pooling need not only group features based on spatial information, but it can also serve to pool nearby features in feature-space. When pooling neighbouring features based on spatial positioning, the pooling reduces the dimensionality of the data from the perspective of spatial size, resulting in a translation invariance. When pooling neighbouring features based on similarity, the pooling reduces dimensionality from the perspective of invariant features, resulting in more complex types of invariance.

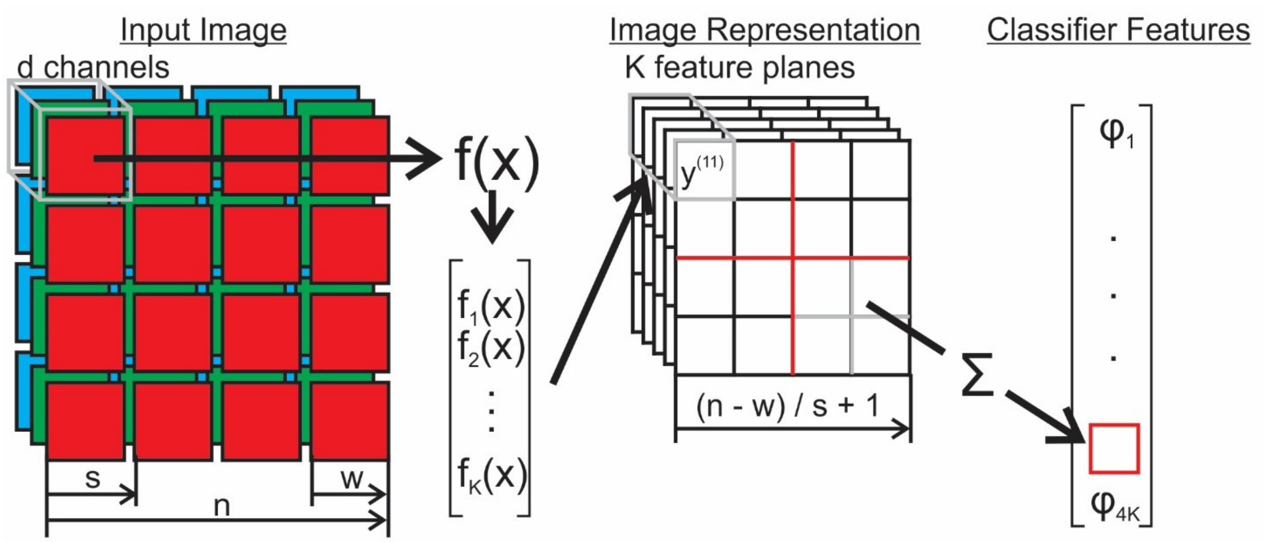


Figure 2.22. Process of feature extraction using w-by-w receptive fields and a stride of s. w-by-w-by-d patches are extracted separated by s pixels each, then they are mapped to K-dimensional feature vectors. These vectors, represented spatially as feature planes, are then pooled over the 4 quadrants in the image and form the feature vector ultimately used for classification.

2.4.5 Comparison of Techniques

An important paper from Coates *et al.* [23] conducted a thorough comparison of unsupervised feature learning techniques and their image classification performance. In addition to a comparison of unsupervised techniques for feature learning, a variety of concepts were compared, including window size, stride, and number of features. To reduce the effect of customizations and complex architectures, the paper compares only single-layer networks using the same training, testing, and classification procedure for all of them. By applying these techniques as a single-layer network, it is easier to determine which technique is best at producing discriminative features for classification. Dimensionality reduction is done with a simple pooling algorithm which sums the feature activations in an image quadrant to generate the feature vector in a process that was described in section 2.4.4. A standard linear classifier, the L2-SVM, is trained with these feature vectors and applied for classification. The tests are performed on CIFAR-10 [128], NORB [129], and STL-10 [23] datasets.

The L2-SVM is just like a standard SVM, but instead of using the standard objective function: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{M} e_i$, the regularization term is updated to use the L2 norm, $\frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{M} e_i{}^2$. This change brings about the properties of the L2 norm in that it is less affected by outliers [130]. This SVM will also be used in this thesis when performing the feature detector comparisons in order to keep the same methodology used by Coates *et al*.

In Coates *et al.*'s work, it is notable that a stride of 1 always produces the best classification accuracy, while increasing strides result in increasingly worse classification rates. Also, more feature nodes result in asymptotically improving performance, with the best results coming from the highest number of nodes tested (1600). Small window sizes performed best (6x6), while larger window sizes showed slightly decreasing performance. This is likely because more features can be represented with smaller windows instead of detectors being wasted to represent the same features translated to different locations within the window. Considering that pooling induces translational invariance during the classification step, it is an inefficient use of the network to encode translated features.

Despite the popularity of probabilistic unsupervised methods for deep learning, it was found that the K-means algorithm learns the best features for classification. This relies on the fact that whitening is performed, because the K-means algorithm cannot learn higher-order correlations on its own. The sparse RBM outperforms the K-means algorithm when whitening is not involved. In fact, when more than 200 hidden nodes are used, the sparse RBM performance is the same with and without whitening, showing that whitening might slightly improve performance but is not necessary for RBMs. It produces consistently high results, compared to the other techniques except the whitened K-means. In a deep network, where more complex representations must be formed, the RBM would perform well with increasing layers since whitening is an uncommon pre-processing step to do between layers.

The structure of the comparison between single-layer networks performed by Coates *et al.* inspires the comparison in this thesis, where unsupervised feature learning using Gaussian filter preprocessing regularization is analyzed in Chapter 3, for their invariance and discriminative feature learning ability.

2.4.6 Encoding Transformations

The GRBM, discussed in section 2.3.1.3, is a higher-order RBM, which is capable of representing more complex information than the visual features described in the previous sections. One of the most relevant applications to this thesis is the GRBM's ability to learn transformations between images, as this will be the primary method by which the model presented in Chapter 4 this thesis will be able to encode motion.

Memisevic`s experimental results [81] as well as those obtained from our own implementation show that this network discovers 2D Fourier components when trained on image pairs that have been shifted, and log-polar variants of the Fourier transform when trained on image pairs with rotation.

When trained on real-world video data, the network discovers a set of filters, which are mostly a combination of shifts and skews. The results show that the GRBM is a useful technique, which follows the free-energy principle, to learn motion representation. They can also be interpreted as learning to behave similar to a complex combination of Reichardt detectors.

The network is also generative, and can apply learned transformations to new images it has not seen before. By presenting two source images, the transformation between the two will be encoded in the hidden activations. Then, by fixing the hidden nodes to these activations, and presenting a new target image, the network will be able to apply the same transformation found between the two source images to generate a transformed version of the target image.

2.4.7 Summary

In this section, sparse RBMs, deep networks, ICA, ISA, and TICA were reviewed. They all learn physiologically-consistent results, and develop features that have proven to be effective for image classification. In fact, the features learned by these techniques are very similar in their first layer, developing Gabor-like filters as the most common representation. RBMs and Deep Belief Networks applied to vision is the most recent approach within the hierarchical architecture. Hinton's RBM is a recurrent network which takes an input and propagates the signal back and forth between its visible layer and hidden layer, reconstructing the visible nodes as well as updating the hidden nodes. Its Hebbian learning based only on local connectivity is more biologically plausible than backpropagation neural networks. This system learns in an unsupervised manner, using the difference between input and reconstruction as a learning signal, and is able to encode statistically relevant relationships between input data. In the case of vision, input data are often pixel intensities and the hidden nodes encode relevant relationships, or features, between those pixel intensities. The sparse RBM constrains the hidden layer to activate infrequently, such that they are forced to only react to important data, resulting in a more efficient encoding. It has been shown that the sparse RBM, learning from real-world images, comes to represent similar feature detectors as V1 in the primate visual cortex, such as oriented edges.

Stacking these RBMs creates a Deep Belief Network, where the hidden nodes of one layer, behave as the visible nodes of the next layer. This stacking allows the network to represent more and more complex visual relationships. Extending the last example, stacking two sparse RBMs that learned from real-world images has the first layer represent feature detectors in V1, and the second layer represent more complex feature detectors in V2, such as contours. The power of stacking these networks extends beyond forming a good basis for visual representation, as it makes the network very scalable and training very efficient. Each layer can be trained on its own, in an unsupervised manner, without requiring the system as a whole to be trained.

Pooling techniques were reviewed which group the same features from neighbouring spatial locations for translation invariance, such as the ones found in CNNs, where max pooling is commonplace. Pooling techniques to group different features from neighbouring locations in feature space for more complex invariances were also reviewed, such as those found in TICA or ISA networks, where L2 pooling is more frequent.

Another important topic that was reviewed is the unsupervised learning of all parts of the network, including the pooling nodes. Learning the invariances for pooling nodes from the data removes the additional assumption of only using translation-invariance and eliminates another element of designer bias. ISA and TICA were reviewed techniques that used real-world images to find dependent components such that they can be grouped together for invariance. These techniques were applied to training deep belief networks, and ultimately found real success by showing an increase in performance over hard-wired translation invariance. Invariance was also learned based on the theory that grouping temporally co-occurring slow-changing features can achieve invariance, thus requiring learning from video. Földiák [120] used a basic temporal averaging function with artificial video data while Einhäuser *et al.* [121] and Zou *et al.* [122] used more complex extraction of temporally slow-changing signals on real-world video. This neurology-inspired idea remains a plausible direction of research as the computational results get closer to mimicking those of biological complex cells.

Results produced by the GRBM were also reviewed. The extension of the RBM model to factor three-way interactions allows the network to realize higher-order concepts, such as the learning of transformations between two sets of images without learning about the images themselves. This ability to generalize shows the power of such a mechanism to learn concepts such as motion in a compact representation. This concept is leveraged for the proposed motion tracking model introduced in Chapter 4.

2.5 Motion Tracking

Motion tracking, or motion estimation, is one of machine vision's most-researched problems. Though smooth pursuit has been discussed in section 2.2.6.3, motion tracking has the same goal without the biological-inspiration or constraints on the models and results. Motion tracking requires the following of one or more objects over time. In traditional machine vision, motion tracking has been accomplished in a variety of ways; with the combination of methodologies being application-specific. Though the proposed research does not need to adhere strongly to the principles of traditional motion tracking, there is a functional overlap with the methods described here. Thus, this section provides an understanding of the aspects of traditional motion tracking such that the original research in this thesis can be adequately compared to them.

There are many difficulties in tracking an object in a video, since detection from frame to frame can be hindered by noise, complex and rapid motion, occlusions, depth rotations, and a myriad of other issues. Object tracking involves many classical machine vision solutions, such as segmentation, feature detection, and feature matching. Theoretically, only one stage is required for motion tracking, and that is object detection in each frame along with the trivial calculation of the positional difference. However, practically speaking, there are enough inaccuracies and computational complexity in detection such that additional stages may be required to impose constraints or to perform filtering. The stage where constraints are imposed facilitates accurate detection, where false detection possibilities are reduced by assumptions or external knowledge. The filtering stage compensates for errors in the detection by using temporal information and other constraints. Some techniques use these two stages distinctly; however, other techniques combine both into one method.

The most rudimentary methodology involves detecting the target initially and then again in each subsequent frame or timestep. This, of course, does not take into account any additional information and requires that the detection is very accurate. The likelihood of false positives in the detection is extremely high for all but the most trivial of tasks. It also may suffer from performance problems, since searching an image for a specific object at every timestep is computationally expensive. This process can be more effective by exploiting the characteristics of real-world motion. Thus, instead of object detection in each full frame, the object is searched for under some constraints. The most obvious motion characteristics are that the object is unlikely to change position drastically nor is it likely that the object will look significantly different in successive frames. As a result, the search area can be reduced to exploit the first characteristic, while detection parameters can be reduced to exploit the second one. These two concepts can reduce computation time while also decreasing the likelihood of false positives.

Background subtraction can be used to isolate foreground objects such that the detection can be constrained to only those areas. In the context of motion, it is a differencing of temporally adjacent frames, where areas that stay the same are static while areas that are different are areas of motion [131]. This differencing is based on a subtraction of intensities; areas without motion have an absolute intensity difference below a certain threshold and areas with motion are above that threshold.

Template matching is the most common approach for general purpose feature matching. This brute force technique searches for a region similar to the object template, found in the previous frame. The most basic type, block matching, attempts to match an image patch in one frame to an image patch in the subsequent frame [132]. A similarity measure can determine how similar the image patches are to each other. For example, squared error can be used to find the best match as shown in Equation (2.35).

$$\left(m_x, m_y\right) = \arg max_{dx,dy} \sum_{x,y} [O(x, y) - I(x + dx, y + dy)]^2 \tag{2.35}$$

where *(m_x, m_y)* is the best match, *(dx, dy)* is the candidate template offset, *O* is the original template, and *I* is the candidate template. The block with the highest similarity measure, can be selected as the match and that transformation can be considered the object motion. This works on the principle that the image patch will not change dramatically from frame to frame. Due to sensitivity to illumination changes, gradients are often used instead of intensity such that the relative intensity differences are preserved [133].

Using features, instead of image patches, can resolve limitations such as searching for large, transparent, or occluded objects [134]. By looking for specific features that are easier to track because of their uniqueness, such as corners or junctions, the search might be more robust.

Given that matching might be inaccurate due to generally incorrect detection, noise, or measurement inaccuracy, using a series of measurements over time can aid in accurate tracking by smoothing out unlikely jitters or inconsistencies in the detection. The Kalman filter [135] is a proven method for exactly this type of problem. The Kalman filter recursively produces an optimal estimate of the values of underlying system variables based on noisy measurements. Provided that the motion variables and noise are normally distributed, the Kalman filter will produce the optimal estimate. The Kalman filter alternates between prediction of the next state and a correction of the state based on the measurement.

The Kalman filter requires a defined linear model which it uses to correct measurements and predict the next state. The model, in the case of object tracking, typically assumes constant velocity or constant acceleration from one timestep to the next. It also assumes two types of noise will be present, measurement noise and process noise. Measurement noise is the idea that the measurement will be inaccurate and process noise is the idea that the model will be inaccurate. The Kalman filter assumes the noise is of a Gaussian distribution. The Kalman filter's next state prediction is based on the current state and the model, while the current state update is based on the last measurement and the previous state. It has been used effectively for motion estimation in the context of video compression [136], image stabilization [137], and depth estimation [138].

To evaluate the current state of motion tracking from the perspective of physiological models, two papers stand out. Mahadevan and Vasconcelos [139] approach object tracking from a biological perspective based on the idea that top-down biasing of the visual saliency mechanism guides the visual attention to track objects. The network alternates between learning features from the bottom-up saliency in the region of the object and top-down biasing of the next frame's saliency calculations by those same features. The most important principle in this mechanism is the idea of discriminant saliency, where visual information is divided into two possible classes: a target class and a background class. The center-surround mechanism behaves as supervision for learning the target, where the visual information in the center component is considered the target and the visual information in the surround component is considered the background. Also, due to the emphasis on visual attention, the system has the ability to automatically select a target via bottom-up unbiased saliency to initialize the tracker. Wang and Yeung [140] implement a motion tracker which uses a deep architecture to aid in the discrimination between the target object and background. By pretraining a multi-layer autoencoder on real-world image data, the encoder part of the network learns compact generic robust features, similar to [53] and [76], which it will later use for classification. An additional classifier is put on top of these features which learns whether the image patch is a background patch or the target object, and this layer is trained online during tracking. Using a constrained neighbourhood, search windows in the surrounding patches are fed to the network to determine the likelihood that they are the target object in the current frame. A particle filtering process aids in the prediction and correction of the tracking. The biologically-inspired methods share an overlap with the smooth pursuit models reviewed in section 2.2.6.3. However, the distinction used here is that biologically-inspired models take traditional motion tracking methods and replace some of the aspects with biologically-inspired techniques or machine learning, while smooth pursuit models aim to provide a computational mechanism by which tracking occurs in the brain.

Chapter 3 Biasing Unsupervised Single-Layer Networks using Gaussian Filters to Learn Invariant Visual Features

The first intended contribution of this thesis is the concept of learning invariant visual feature representations based on temporal information. This is based on the idea that an object in a visual receptive field will change over time and those changes can represent certain common transformations. The same object will create slightly different projections on the receptive field over time, and a recognition framework must be tolerant to those slight changes in order to correctly recognize the object despite its configuration. This makes video a viable source from which features that are resistant to common transformations can be learned, since they are already present in video and linked by temporal proximity. The use of static images to learn invariant features was abundant in literature through methods that induce dependence when feature detectors are being trained such that many feature detectors encode similar features under different transformations. Despite the introduction of many techniques and aesthetically appealing results, the reasoning behind, and the quantification of their behavior was largely uncontested. The evaluation and in-depth characterization of existing methods of invariance learning was an area that was lacking in literature. As a result, the actual first contribution of this thesis is the thorough analysis of basic neural network biasing techniques to learn invariant visual feature representations, using both real-world static images and real-world video, in an unsupervised manner. As feature invariance is commonly credited for improved classification performance, specific attention will also be paid to analyzing the relationship between invariance and classification performance.

Feature invariance is hardcoded in frameworks such as the CNN, and has been used to great success. It is hardcoded in the form of translation invariance by allowing the same feature to appear in adjacent positions and a pooling produces an unchanged or similar response despite the position.

There has been a rise in biologically-consistent results developing from unsupervised learning techniques trained on real-world image and video data. Simple cells represent feature selectivity whereas complex cells specialize in feature invariance. ICA and sparse RBMs have accomplished learning features from real-world image data similar to the response of simple cells in the Hubel-Wiesel architecture, but far less research has gone into learning invariances similar to complex cells.

There have also been significant performance increases in image representation and classification through the advancement of techniques involving unsupervised learning and deep architectures. The importance in this performance increase is that these learning mechanisms learn statistically-relevant features directly from the data, which are then used for classification with a more traditional supervised method. Interestingly, when exposed to large amounts of real-world image data, modern unsupervised learning techniques have produced physiologically-consistent results, learning features similar to neurons in V1. Furthermore, the deep learning movement has produced state-of-the-art results on object classification tasks using stacked layers that are greedily trained in an unsupervised manner layer-by-layer.

Many of the different techniques discussing improved methods use multi-layered models, and are able to justify the performance on their layer-by-layer techniques by looking at classification statistics of the whole model. A more effective way to assess these methods and their properties is to test them on single layer networks, where the influence of network architecture is minimized and the actual performance of the learning mechanism can more easily be isolated and compared. Coates *et al.* [23] structured their comparison similarly by comparing single layer networks and their classification performance on standard datasets under a variety of parameters such as stride, receptive field size, and number of hidden nodes. That study outlined the efficacy of several single layer techniques at learning discriminative features from raw pixel data. Modeled after a study of the basic single-layer network, our current research aims to compare different regularization and preprocessing techniques also on a single layer in order to boost classification performance through generating better features. In addition, the properties of these features at higher levels, which are more difficult to visualize and interpret, may lead to further understanding of their role in object recognition.

An important aspect of learning is the ability of a system to be robust enough to generalize between slightly different versions of the same stimulus, but specific enough to discriminate between the desired stimulus and other stimuli. At the heart of this particular balance lies the concept of invariance. Typically, unsupervised learning techniques will produce features which are not particularly invariant to affine transformations. Affine transformations are typically sufficient to characterize the possible transformations that can occur to a feature in the real world, so increasing a feature's tolerance to such transformations is considered making the feature more invariant and ultimately a more robust data point for classification.

The Hubel-Wiesel architecture introduces the idea of a hierarchical vision system that has simple feature detectors connected to complex nodes which are invariant to a small degree of transformation. The continued hierarchical alternation of these two types of nodes gradually achieves an invariant representation. Architectures such as the Convolutional RBM (CRBM) [141] or the CNN [107] use hard-wired translational invariance to achieve robustness. By tolerating small positional changes of the same feature at increasing levels of the hierarchy, the network learns to detect objects in different positions and configurations than the training examples it was exposed to. Though efficient and particularly effective, hard-wired translation invariance limits the network to tolerating only one type of transform. With the effectiveness of unsupervised learning methods at learning visual features from image data, it is a natural progression to also learn invariances from the same data, which is attempted in this chapter.

Regularization is the process of biasing a function based on external information with the goal of producing more desirable results. RBMs [77] and autoencoders [22] have benefitted from regularization to induce sparsity, resulting in the production of more discriminative features, such as physiologically-consistent Gabor-like features when trained on real-world image data. TICA [79] and topographic RBMs [112] use regularization to produce topography in the learned features. Le *et al.* [22] created a large scale deep network that used neighbourhoods of features found using TICA to achieve invariance en route to state-of-the-art results on a variety of object recognition benchmarks. Capturing slow-changing features in real-world video through low-pass filtering of activations on sequential frames of video produces features that are invariant to transformations [120]. Zou *et al.* [122] simulate fixations to track objects over time. Slow-changing features are learned for the purpose of invariance, and lower error rates are produced in image classification tasks. These results show that there can be an increase in performance over hard-wired translation invariance when invariant features are learned from the data, which gives a real reason to explore and quantify methods to achieve this result.

Gaussian filters are at the core of many machine vision techniques, and serve as a desirable function for many purposes. They are able to reduce noise or high frequency effects, incorporate information from neighbouring regions, and soften edges depending on how, and to what, they are applied. In unsupervised learning, they have been considered for inducing topography, falling under the idea of incorporating information from neighbouring regions [112]. Temporal low-pass filtering has been used to learn slow-changing features [120], therefore, Gaussian filters are a viable candidate for implementing the low-pass filter. Generally speaking, incorporating additional information and filtering can produce desirable effects for the purposes of improving feature learning, and the Gaussian filter provides exactly those characteristics. As a result, they are a compelling topic to link together different biasing techniques in unsupervised learning.

This chapter will characterize the effectiveness of regularization and preprocessing methodologies, specifically based around the Gaussian filter, at learning discriminative-yet-invariant features by a set of metrics as well as classification performance on standard datasets. It is often considered that enforcing independence among learned features allows the limited number of hidden nodes to represent the most diverse set of features while reducing redundancy and maximizing representational power. This chapter will challenge that notion by comparing features learned by enforcing independence against features learned by incorporating neighbouring information. The features will be learned from the Hollywood2 video dataset [142], the static CIFAR-10 [128], STL-10 [23], and COIL-100 [125] image classification datasets, and the MNIST handwritten digits dataset [102]. The learned features will be used for image classification as a measure of how effective they are, thus they are also tested on the CIFAR-10, STL-10, COIL-100, and MNIST datasets; they are not tested on Hollywood2, because it is not an image classification dataset. Training on one dataset and testing on another in the visual domain has been shown to work effectively [122]. To reduce the variability when comparing different unsupervised learning frameworks, an RBM with fixed hyperparameters is used as a singular learning architecture on which the different regularizations are tested.

3.1 Compared Biasing Methods

The objective of this part of the thesis is to investigate different methods of applying Gaussian filters to induce different effects during unsupervised learning in an RBM. The compared biasing methods will be: topography induction, temporal low-pass filtering, and input blurring, and will be compared to regular RBM (not biased).

When training the RBM in a batch, the weighted sums in Equations (2.10) and (2.11) are performed by matrix multiplication. The visible inputs matrix, as illustrated on the left-hand side of Figure 3.1, contains all of the training data to be used in the current batch, where each row represents a training pattern, corresponding to an image patch in this thesis, and each column represents a visible node. This means that the pixels values from an image patch are flattened into a row vector. The weight matrix contains the weights connecting each visible node to each hidden node, with each row representing a different visible node and each column representing a different hidden node. The hidden activation matrix, as illustrated on the right-hand side of Figure 3.1, is calculated by multiplying the visible matrix and the weight matrix, resulting in a matrix of activations with each row representing a pattern and each column representing a hidden node. Figure 3.1 shows how a Gaussian filter is applied in the hidden activation matrix and the visible inputs matrix for each of the three biasing methods which will be compared.

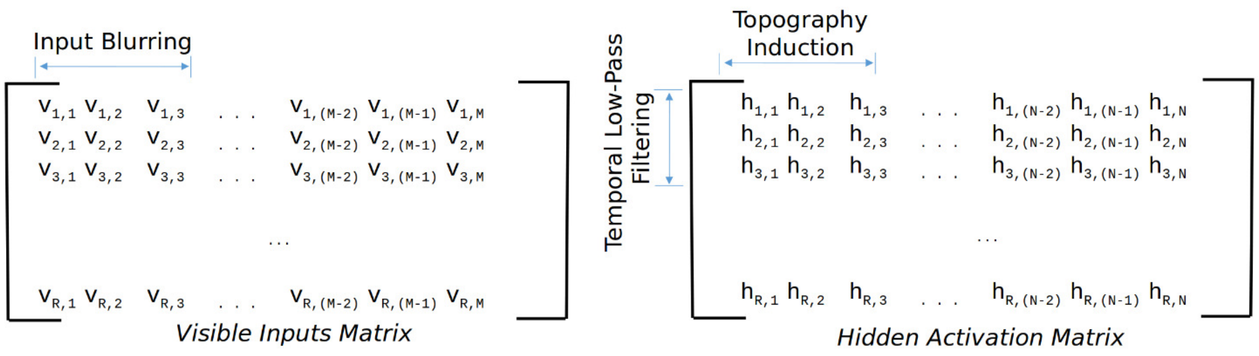


Figure 3.1. Application of convolution, where h are hidden node activations, v are visible inputs, M is the # of visible nodes, N is the # of hidden nodes, and R is # of patterns. Gaussian filters for input blurring is applied to the visible inputs matrix (left), while Gaussian filters for topography induction and for temporal low-pass filtering are applied to the hidden activation matrix in the shown directions (right).

The 1-dimensional Gaussian filter is defined by Equation (3.1).

$$g(x) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot e^{-\frac{x^2}{2\sigma^2}} \tag{3.1}$$

The 2-dimensional and 3-dimensional Gaussian filters are defined by Equations (3.2) and (3.3), respectively.

$$g(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.2}$$

$$g(x,y,z) = \frac{1}{\sqrt{8\pi^3}\sigma^2} \cdot e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \tag{3.3}$$

The discrete kernels are sampled from the above equations. The purposes of the Gaussian filter in this thesis are conceptually varied, despite obvious similarities. In the case of the topographic RBM, the Gaussian filter incorporates information from local hidden nodes to induce a dependence which influences nodes to develop properties similar to their neighbours. In the case of temporal activation smoothing, the Gaussian filter serves to remove fast-changing filter responses such that only constant features in video, despite mild transformations, cause activation. For the purpose of image blurring, the Gaussian filter serves to soften edges such that the network learns blurrier features. Note that these filters are normalized such that their gain stays at 1 whenever they are applied in this thesis.

The neighbourhood sizes were kept small, 1x3, 3x3, or 3x3x3, depending on the biasing technique. This was done to reduce computational complexity, as the amount of processing time for each experiment would increase exponentially with kernel size; given that so many experimental evaluations were done, the time constraints forced a smaller kernel size. Though these kernels are small, they do allow a differential between the center element and the neighbouring elements. This differential is what is important in determining the effect of the compared methodologies, and ultimately, determining the importance of incorporating local information instead of enforcing independence in feature learning.

3.1.1 Regular RBM

A sparse RBM without any additional regularization is used as a control and will be compared as a baseline. This will be referred to as the regular RBM. The sparse RBM is often used as the basic method upon which deep networks are built, and is often used for feature extraction purposes [143]. By using this method as a baseline, it will allow us to determine whether improvements can be made on this method and infer that an improvement in one layer could be effective in a multi-layered deep network also.

3.1.2 Topography

Topographical structure can be found in the visual cortex [144]. Retinotopic maps contain a direct mapping from adjacent retinal positions to adjacent neurons, such that the entire visual field is represented by a continuous map in the cortex, such as V1 and LGN. In other areas, such as V2, the retinotopic map is divided into two, such that one area corresponds to the upper half of the visual field and the other corresponds to the lower half. Other areas have more complex divisions and represent more complex features. For example, orientation maps exist where neurons respond to visual stimuli of different orientations [80]. Orientation maps typically have a topography, where similar orientations will be adjacent such that the map varies smoothly across orientations along the adjacent areas in the neuron collection.

The primary theory for this type of organization is that it is more efficient to have highly interconnected neurons be near to each other [145]. There are significant costs to long distance connections. For example, there is a high metabolic cost to create a long distance connection, and high latency during communication. For pooling similar areas together for invariance, such as the presumed nature of the complex cell, having the cells that are pooled be nearby makes sense since they fire at similar times and connect to a common neuron. Also, the idea that lateral inhibition may increase selectivity by suppressing similar responses requires an ordering of neurons. In the interest of efficiency, it is likely that self-organization contains the additional constraint of minimizing wiring length or latency related to strong neuronal connection such that topographic maps emerge.

Topography is induced in the RBM by regularization. By ordering the hidden nodes for a 1D topography, or arranging the hidden nodes in a grid for a 2D or 3D topography, neighbouring nodes can be determined. Applying a Gaussian filter to hidden node activations at each example, each node incorporates information about its neighbouring nodes. Applied during learning, adjacent nodes develop slightly different feature responses that gradually vary across the arrangement.

Assuming batch training, a Gaussian smoothing filter is applied among adjacent hidden nodes exposed to the same pattern in the activation matrix. The positive phase activations of the hidden nodes, $h^{(k)+}$, are calculated by Equation (2.10) when presented with the kth visible pattern as v, and then modified by Equation (3.4) to induce topography as found in [112].

$$\hat{h}_j^{(k)} = \sum_{n=1}^{N} h_n^{(k)+} \omega(j,n) \tag{3.4}$$

where $\hat{h}_j^{(k)}$ is the topography-induced positive activation of hidden node j at pattern k, $h_n^{(k)+}$ is the positive phase activation of hidden node n at pattern k, and the neighbourhood function, ω, is a set of fixed neighbourhood weights which controls the impact of the surroundings on each activation. ω is set to a Gaussian function, sampled from Equations (3.1), (3.2) or (3.3), depending on dimensionality of the grid that the hidden nodes are arranged in. A 1x3 kernel, 3x3 kernel, and 3x3x3 kernel are used for the 1D, 2D, and 3D topographies, respectively.

3.1.3 Temporal Low-Pass Filtering

Invariant representations may also be learned from video. Prior work has indicated that temporal low-pass filtering on activations during learning of transformation sequences gives rise to invariant features [120, 146].

Conceptually, a single neuron may be activated by a visual stimulus generated by a real-world object. Assuming that there is a temporal decay of activation, there will be some effect of the previous activation on the next one instead of the activations being completely independent. In the time interval that the previous activation has not decayed to 0, it is likely that the visual stimulus will be transformed slightly. The same neuron will then be learning to respond to both the original stimulus as well as the transformed stimulus, learning a more invariant representation. It is unlikely that another object will stimulate that same neuron in the short window of time, since it is unlikely that a completely different object will occupy the same receptive field shortly after the previous object. Due to real-world behaviour often being constrained by temporal and spatial limitations, this is an acceptable line of research for invariance learning.

There is evidence that neurons may learn spatial correlations via temporal continuity. The temporal properties of learning might be beneficial to learning continuity in the real-world. The electrochemical transmission process during learning does not behave discretely during learning and has a period of time over which learning occurs [147, 148]. This may serve as temporal averaging if neuronal activity changes during the learning interval. Also, the prolonged firing of inferior temporal neurons after the stimulus is no longer shown [149] may be a sufficient time interval to provide the capability of involving previous activations into the learning of the current activation.

Since consecutive patterns, or individual rows in the visible inputs matrix, are the same image patch in consecutive frames extracted from a video, a temporal continuity effect is induced during batch training by applying a Gaussian filter to the hidden activation matrix perpendicular to the way it was in the topography organization. This corresponds to blurring among the same hidden node exposed to temporally-adjacent patterns, as shown in Equation (3.5). As a result, a temporal continuity effect is induced. This transposes the ideas presented in the previously mentioned papers within the single-layer RBM framework used in this thesis.

$$\hat{h}_j^{(k)} = \sum_{r=1}^{R} h_j^{(r)+} \omega(k, r) \tag{3.5}$$

where $\hat{h}_j^{(k)}$ is the temporal low-pass-induced positive activation of hidden node j at pattern k, $h_j^{(r)+}$ is the positive phase activation of hidden node j at pattern r, estimated with Equation (2.10), and this ω represents the Gaussian weighting of the activation that different patterns produce on the same hidden node, which is a type of temporal low-pass filter. Since it is 1-dimensional, ω can be represented by Equation (3.1). A kernel width of 3 is used.

3.1.4 Input Blurred

Input blurring can be considered in a number of ways; resolution reduction and detail ambiguation are considered the most relevant for this thesis. The CNN and other related Hubel-Wiesel architectures effectively implement both, where resolution is reduced at each level of the model hierarchy as well as inducing ambiguity of position due to transformation invariance of pooling nodes. In those architectures, dimensionality is explicitly reduced and ambiguity of position is explicitly induced while feature selectivity is still high. Biologically, this hierarchy is far more information-rich, and can reduce dimensionality during upwards propagation for recognition while keeping sufficient spatial information to identify precise position.

In this thesis, input blurring uses a Gaussian low-pass filter, performing a type of resolution reduction. In the process, feature ambiguity is also increased, yet without the ability to have high selectivity since there is no additional information. Though a loss of resolution is detrimental for many applications, in the case of training, such a property may actually be a benefit to learning features that are more invariant. This method could fall under preprocessing, however, it is addressed as a separate method used for comparison. Each image patch is blurred using a Gaussian filter with kernel width 3 and varying standard deviation (σ), using Equation (3.6), before being passed as inputs to the RBM.

$$v_i^{(k)} = \sum_{m=1}^{M} v_m^{(k)} \omega(i,m) \tag{3.6}$$

where $v_i^{(k)}$ is smoothed visible node i at pattern k, $v_m^{(k)}$ is a neighbouring visible node m at pattern k, and the neighbourhood function, ω, is the Gaussian filter. In the case of a 2-dimensional image, as is the case in this thesis, Equation (3.2) is used for ω with a 3x3 kernel.

3.2 Experiment Design

The main evaluation of performance will be the classification accuracy of the learned features from each methodology on different datasets. A variety of datasets are used, covering different types of visual recognition, all of which are described in section 3.2.2. The general procedure is that the methodologies will be trained on image patches from various datasets, as described in section 3.2.4. The learned features will then be tested on the same or different datasets by evaluating the accuracy of a classifier when using the responses of those features as inputs; this process is described in section 3.2.5. Finally, an invariance score will also be calculated for the features learned by each method to compare their resistance to affine transformations, as described in 3.2.6.

3.2.1 Implementation

The RBMs were created and trained using Pylearn2 [150], with custom modifications to implement the regularization techniques. GNU Octave [151] was used to implement the testing procedure, including the convolution and the L2-SVM, which was strongly based on the publicly-available code from Coates *et al.* [23], in order to keep as many things in common as possible.

3.2.2 Datasets

A variety of datasets were used to both satisfy the training constraints and to test the methods' performances on different visual domains.

The Hollywood2 action dataset [142] was used for training. It is a real-world video dataset containing colour video clips taken from a variety of Hollywood films; five examples are shown in Figure 3.2. The sizes of the videos are not fixed, and range between 480x224 and 720x576. The dataset contains enough real-world scenery, image variation, and motion to suit the training algorithms that use static images (regular, topographic, input blurred methods) as well as the ones that make use of video (temporal low-pass method).

Figure 3.2. Examples from Hollywood2 dataset.

CIFAR-10 [128] is composed of real-world 32x32 colour images containing labelled objects belonging to 10 different classes, with a total of 50,000 training images and 10,000 testing images; the separation of training dataset and testing dataset is important here since the classifier is trained on the training dataset and tested on the testing dataset. Samples of CIFAR-10 are shown in Figure 3.3.

Figure 3.3. Some examples of images in CIFAR-10, with labels of dog, automobile, and airplane (Left to Right).

STL-10 [23] is similar to CIFAR-10 with 10 classes, but contains 96x96 colour images. There are 5000 labeled training images divided into 10 folds, in addition to 8000 test images also divided into 10 corresponding folds. Training and testing occurs on each fold with the corresponding subset of images, and the mean performance on all of the folds is the reported result. Some examples are shown in Figure 3.4.

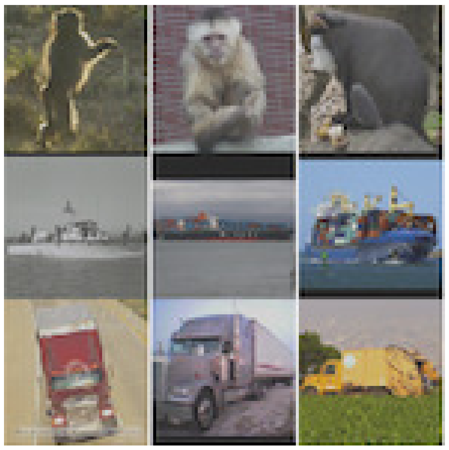


Figure 3.4. Examples images from STL-10, with different examples of monkeys, ships, and trucks (Top to Bottom).

The COIL-100 [125] dataset contains household items photographed at different angles against a black background. Each of the 100 objects has grayscale images, of 32x32 pixels each, taken under rotations of 15 degree angle increments about the vertical axis, comprising 24 images per object. The testing procedure requires that only the 0, 90, 180, and 270 degree images are used for training. Classification accuracy is determined by using the remaining images to predict the correct object class. Figure 3.5 shows each of the 100 objects in COIL-100, from one of the 24 different angles they are photographed in.

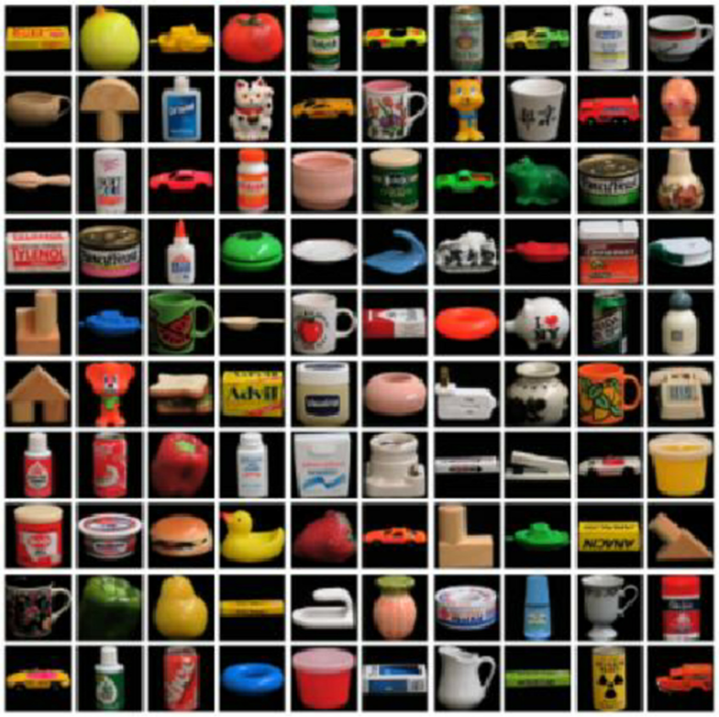


Figure 3.5. Each of the 100 objects in COIL-100, shown from one of the 24 orientations available.

MNIST [102], as discussed in section 2.3.2.1, is a database that consists of handwritten digits. 50000 28x28 black and white images are used for training, while 10000 are used for testing. On this dataset, an improvement of 0.2% is considered statistically significant [152]. Examples of MNIST images are shown in Figure 3.6.

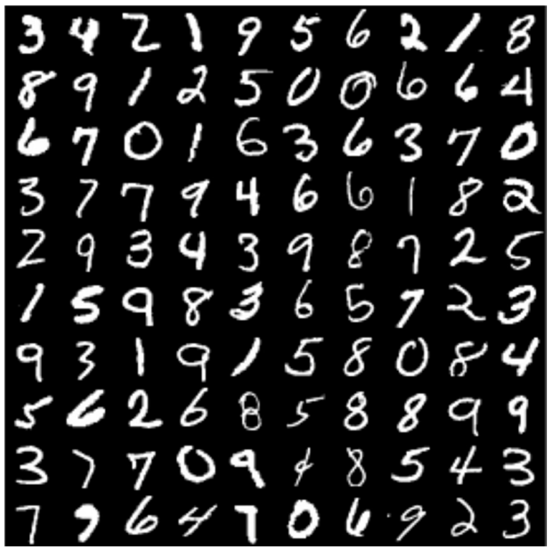


Figure 3.6. Selected digit images from the MNIST dataset.

3.2.3 Preprocessing

Instead of the raw pixels being fed to the networks directly, the colour patches extracted from the images are contrast normalized and whitened, as these are common techniques to reduce redundant information [23].

Contrast normalization is performed by subtracting the patch mean from within each patch and then dividing each pixel by the standard deviation of the pixels in the patch. To follow a procedure commonly used in deep learning [53], the dataset is whitened to remove additional redundancies such that they do not interfere with the network learning important features. However, there is evidence that whitening provides no tangible benefit when training RBMs with large amounts of hidden nodes [23]. As some of the results in this thesis involve using very few hidden nodes, and the training procedure is kept the same across all experiments, whitening is employed to get results of the highest accuracy.

3.2.4 Training

The initial comparison uses the Hollywood2 action dataset for training, such that the different methodologies can be contrasted, including temporal low-pass filtering. To efficiently use the video dataset, some sort of visual attention must be implemented. With real-world images, each individual patch is different, so if one patch is not particularly informative, it will not affect the training for more than being just one sample. However, with real-world video, if a textureless location is used, and it does not contain any motion, such a location may not have any useful information for an indefinite amount of weight updates. As a result, a mechanism to find motion is used in the form of a background subtraction between adjacent frames, selecting the position with the highest difference, and using the image patch at that location as a training example. The sequence lasts for 5 frames until a new position is found through the same process. Any method could be used to find patches of interest; the described method was chosen to keep the training time low by finding valuable training data, and to keep the computational complexity low by making it a simple subtraction and sorting algorithm. Sequential frames are consecutive training examples such that the temporal low-pass filter can operate as described. The network was trained for 300 epochs, on 10,000 5-frame sequences of the selected patches, divided into batches of 100. The patch size is typically 8x8, except in one experiment where it is reduced to 6x6 for reasons explained in the text.

The other datasets are also used for training for specific experiments to determine how well the compared techniques perform on different visual data, however, since they are static images, the temporal low-pass method is excluded from these comparisons. When using static datasets for training, the network was trained for 300 epochs on 50,000 randomly-selected patches from the dataset.

Colour images are represented in three channels for each pixel: red, green, and blue. Also, since the networks have no knowledge of spatial structure or colour, the image patches are unrolled into a 1-dimensional vector before being passed to the input layer of the RBM.

3.2.5 Testing

To determine how effective the learned features are, they are used as the descriptors in a visual classification task. The trained networks are tested on different datasets: CIFAR-10, STL-10, MNIST, and

COIL-100. Classification accuracy is reported according to the standard testing procedure for each dataset, explained in section 3.2.2.

The classification procedure follows the method used by Coates *et al*. [23], described in section 2.4.4 and illustrated in Figure 2.22, where features are extracted using the trained network and pooled by summing feature activations in each quadrant of the image. Just as in that paper, an L2-SVM is used for classification.

3.2.6 Invariance Metric

Trying to compress the behaviour of a network into a simple metric is difficult. The performance of an individual feature's invariance properties can be measured for rotation and translation quite easily. The rotation selectivity can be measured by applying the optimal stimulus at different orientations. When graphed, the peak will be at the optimal stimulus. The narrowness of the peak specifies how selective the feature is to the rotation of its optimal stimulus. The translational invariance can be measured by applying the optimal stimulus at different positions. When graphed, the broadness of the peak specifies how invariant the feature is to the translation of its optimal stimulus, as depicted conceptually in Figure 2.21. Scaling is not considered, because the learned features come to represent edges, and there is no logical way to measure scaling as the optimal stimulus will always be a high contrast line.

The metric proposed by Goodfellow *et al.* [153] simplifies each network's invariance properties into a comparable metric. The idea is to compare a feature's activation when presented with transformed optimal stimuli versus random stimuli.

The local trajectory T(x) is a set of stimuli that are semantically similar to some reference stimulus x:

$$T(x) = \{\tau(x, \gamma) |\ \gamma \in \ \Gamma\} \tag{3.7}$$

where $\tau(x, \gamma)$ transforms a stimulus *x* into a new stimulus, and γ is a transformation which comes from a set of possible transformations, Γ. For example, these transformations could be all translations within 5 pixel radius, or rotations within 45 degrees.

The robustness, L(j), of a hidden node, j, can be calculated by the firing of a hidden node when applied to local trajectories around inputs which maximally activate that node:

$$L(j) = \frac{1}{|Z|} \sum_{z \in Z} \frac{1}{|T(z)|} \sum_{x \in T(z)} f_j(x) \tag{3.8}$$

where $f_j(x)$ is the activation of hidden node j when presented with stimulus x, Z is the set of inputs that produce a high activation from that same hidden node, and T(z) is the set of local trajectories around input z.

The global firing rate, G(j), is the firing rate of a hidden node when applied to stimuli drawn randomly from a distribution of possible inputs defined for the test. Finally, the invariance score from [153], S(j) is calculated:

$$S(j) \ = \frac{L(j)}{G(j)} \tag{3.9}$$

This calculation represents the robustness of the hidden node divided by the global firing rate, where L(j) indicates how robust hidden node j is to local transformations of the optimal stimulus, while G(j) ensures that the hidden node is selective and not always active. A higher invariance score, S, indicates that the hidden node is more invariant.

Since the network may not dedicate all of its resources to encoding invariance information, only the *p* top-scoring nodes are used in calculating the average score for the network, while the remaining (1-*p)* samples are discarded.

3.2.7 Visualization

To visualize the learned feature detectors, the hidden nodes of the network are arranged into a 2-dimensional grid, where each element in the grid represents a 2-dimensional patch of pixels that hidden node responds best to. The weights of each node are reshaped into a 2-dimensional patch of pixels, where each pixel is shaded according to the value of the corresponding weight. A dark pixel indicates a strong negative weight, a light pixel indicates a strong positive weight, and a gray pixel indicates no weight. Effectively, this patch represents the type of feature that a feature detector responds strongest to. For example, a vertical feature node responds best to vertical edges, a coloured feature node responds best to the displayed colour, and a patterned feature node responds best to the displayed pattern. The entire image displays all such patches in a grid, as displayed in Figure 3.7.

3.3 Results and Discussion

Experiments were carried out with regular RBMs using the methodologies detailed in section 3.1: regular, 1D, 2D, and 3D topographies, temporal low-pass, and input blurred. All networks have the same sparsity of 0.01 and weight decay of 0.0002, with no momentum term and no decaying of any hyperparameters. An 8x8 colour receptive field size was used in all tests except where otherwise stated. The 8x8 field is sufficiently small to yield good classification results and sufficient feature coverage, while being large enough that optimal stimuli can be appropriately translated and rotated to generate enough data for the invariance score metric. Any other parameters specific to each technique are outlined in the results.

In the case of the 3D topography, to keep the same number of hidden nodes in each dimension, a different number of total hidden nodes than the 1D and 2D topography had to be chosen for comparison. As a rule of thumb, the largest cube number that is smaller than the comparison number of hidden nodes is chosen as the number of hidden nodes for the 3D topography. This amounts to 216, 343, 512, 729, 1000, and 1331 nodes for the 3D topography corresponding to 225, 400, 625, 900, 1225, and 1600 nodes for the 1D and 2D topographies, respectively.

Examples of features learned by the regular RBM and the topographic RBMs, with 8x8 patches of 3 colour channels resulting in 192 input nodes, and 400 hidden nodes (343 for the 3D topography), are shown in Figure 3.7. Examples of features learned by the temporal low-pass RBM and input blurred RBMs are shown in Figure 3.8. All features are re-arranged into 2-dimensional grids for display purposes, as described in section 3.2.7. The regular, temporal, and input blurred RBM have no ordering and no topographical structure, and are visualized as a 2D grid because a 1-dimensional display would take up too much horizontal space. The 1D topography is visualized as a 2D grid for the same reason, even though it is ordered. The 3D topography is visualized as a 2D grid due to difficulty visualizing the 3-

dimensional relationships between features. These features were learned by exposing the models to 300 epochs of 50,000 8x8 colour image patches from the Hollywood2 dataset, according to the process described in section 3.2.4.

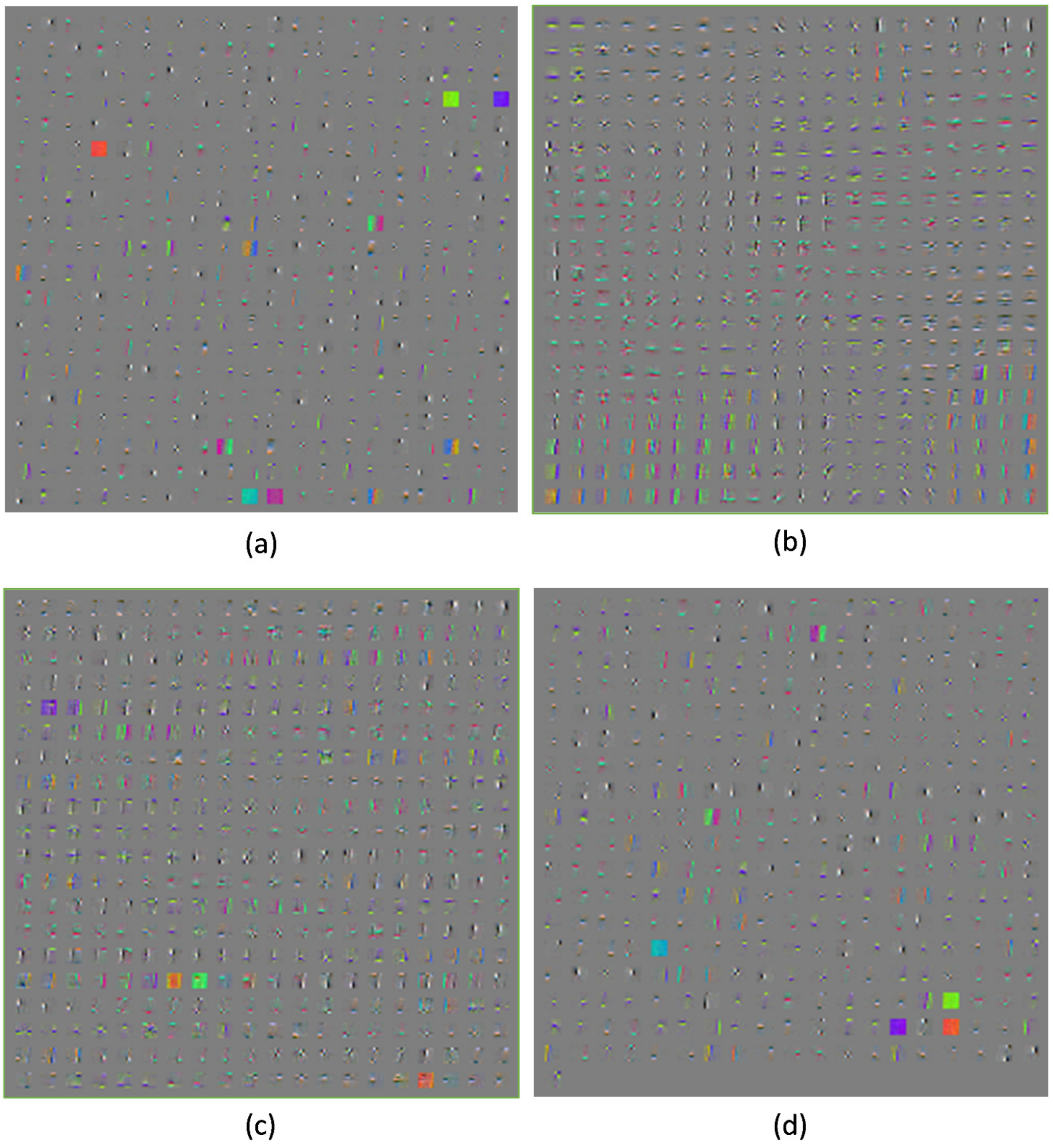


Figure 3.7. Visualization of the features learned by (a) regular RBM (400 features corresponding to 400 hidden nodes), (b) 400x1 1D topography (400 features), (c) 20x20 2D topography (400 features), and (d) 7x7x7 3D topography (343 features).

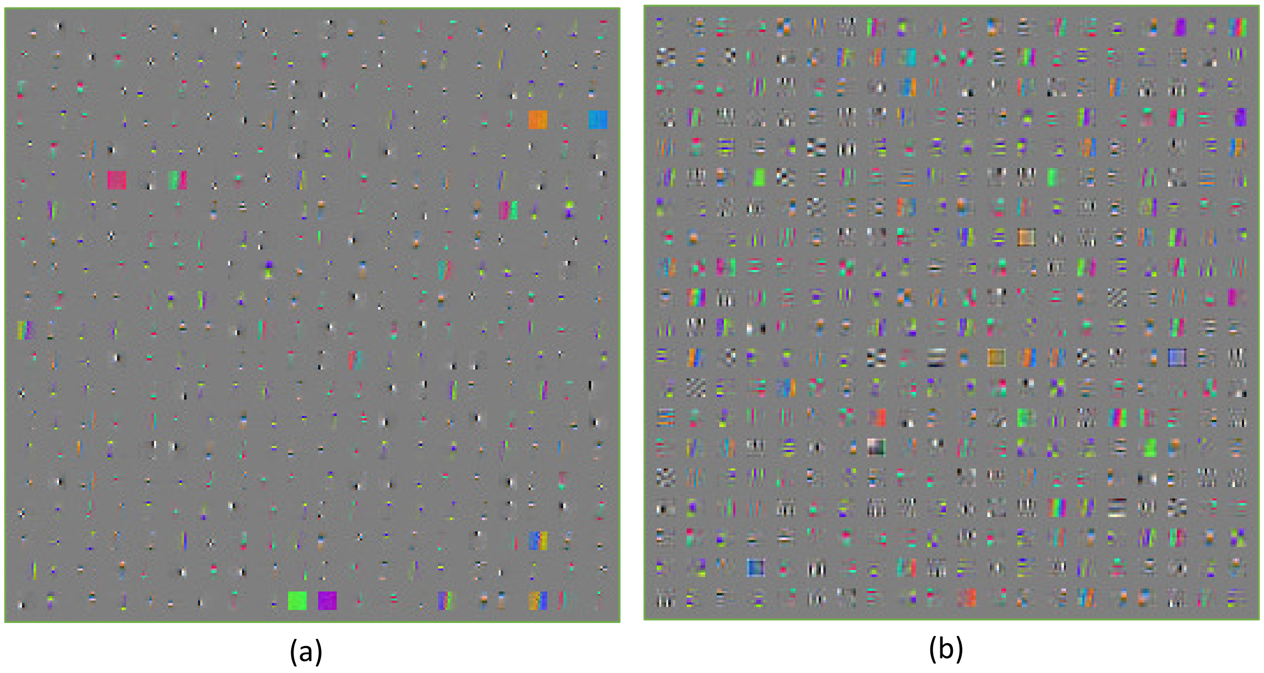

(a) (b)

Figure 3.8. Visualization of the 400 features learned in (a) temporal low-pass RBM and (b) input blurred RBM.

The 1D and 2D topographical structure of the hidden nodes contains smooth variations among neighbouring nodes, as seen in Figure 3.7, where topography is induced by a Gaussian filter on the hidden node activations during training. The 3D topography, also shown in Figure 3.7, produces similar results to the other topographies, however there is difficulty visualizing the continuity due to the flattening of the 3D structure into a 2D image. It can be seen that the topography features are larger and less localized than the regular RBM (Figure 3.7a) due to the dependence on their neighbours and providing continuity by overlapping the features of adjacent nodes. There are also fewer isolated solid colour feature detectors, as the importance of encoding oriented edges creates more of those feature detectors and influences the neighbouring nodes enough that they develop similar detectors. Since none of the hidden nodes can independently develop as a feature detector, the pressures of their surroundings prevent the less common features from emerging when topographic blur is applied. It would seem that this gives the network less representational power, but the classification results discussed in section 3.3.1 will show otherwise.

The temporal low-pass filtered RBM produces very similar features to the regular RBM, as seen in Figure 3.8a compared to Figure 3.7a. Given that all of these networks start with the same initialization and go through the same sequence of patches, it is easy to see that the temporal low-pass filter gets a similar learning signal to the regular RBM, since many of the features are similar in similar positions. However, the edge features are a little less refined and less localized. It is difficult to say whether the coloured features in one are better than the other; they both develop solid colours, but the representational power of the network can only be measured quantitatively. This comparison will be performed below.

The features learned by the RBM with input blurring shows many more high frequency features with less localization, as seen in Figure 3.8b. It is the most different of the compared methods, with feature detectors that often occupy the entirety of the patch, with coloured features appearing frequently instead of just edge detectors. The topographical RBM also has high frequency features with less localization, but they look less clean and isolated than when input blur is applied. While topographical RBM features are less pleasant to look at than the regular and input blurred methods, it may indicate that it has learned a more diverse and complex set of correlations.

3.3.1 Classification

Figure 3.9 shows the different biasing methods and their classification accuracies relative to the number of hidden nodes in the network, when trained on Hollywood2 and tested on CIFAR-10. Within each method, the sigma for the Gaussian function used to bias the network during training with the best classification accuracy at 1600 nodes is selected as the representative for that method, determined by evaluations of sigma between 0.25 and 2.0. The testing procedure is detailed in section 3.2.5.

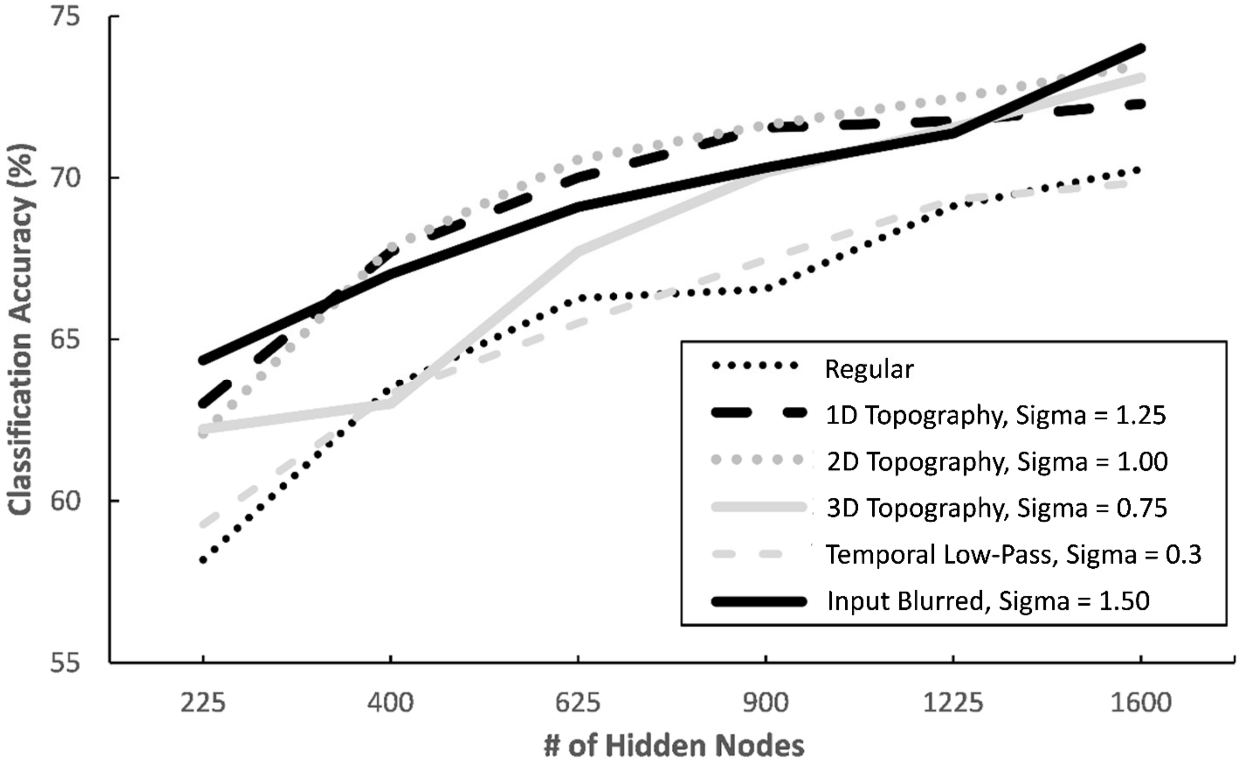


Figure 3.9. Comparison of the classification accuracy on the CIFAR-10 test set vs the number of hidden nodes, with the sigma of the best Gaussian biasing parameters selected for each method.

As with the regular RBM, the classification performance increases with more hidden nodes for all of the techniques. Overall, the input blurring and the topographical methods show a constant improvement over the regular sparse RBM and temporal low-pass filtered RBM at each node count, which shows a larger percentage-wise improvement at the lower node counts than the higher node counts.

The selection of the sigma parameter in the Gaussian filter is an important one. The classification results of each of the techniques with varying sigma values is shown in Figure 3.10 for 225 nodes (216 nodes for the 3D topography), and Figure 3.11 for 900 nodes (729 nodes for the 3D topography).

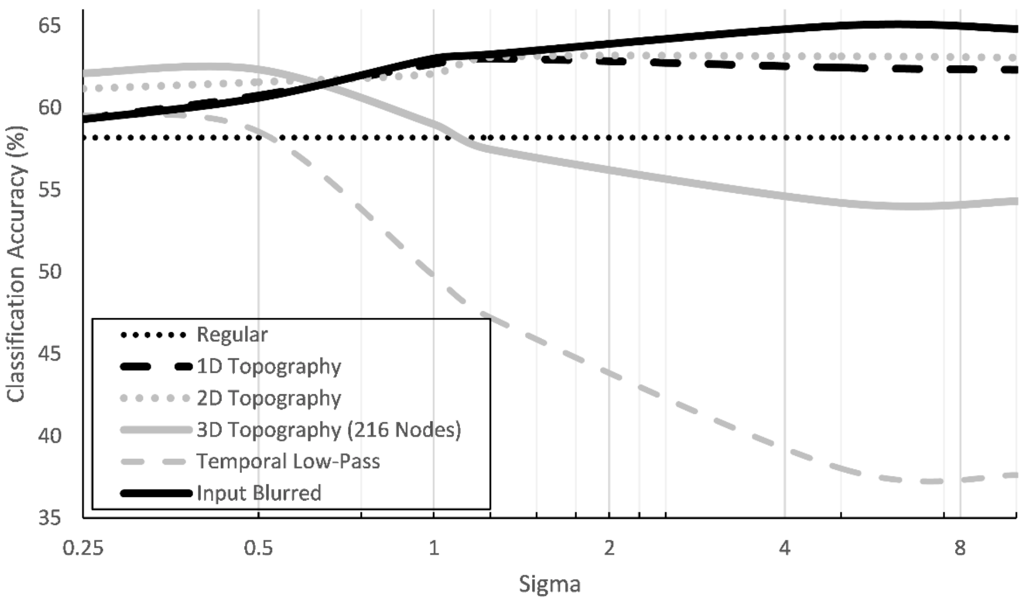


Figure 3.10. Classification accuracy on the CIFAR-10 test set vs sigma of the Gaussian function for each technique at 225 nodes.

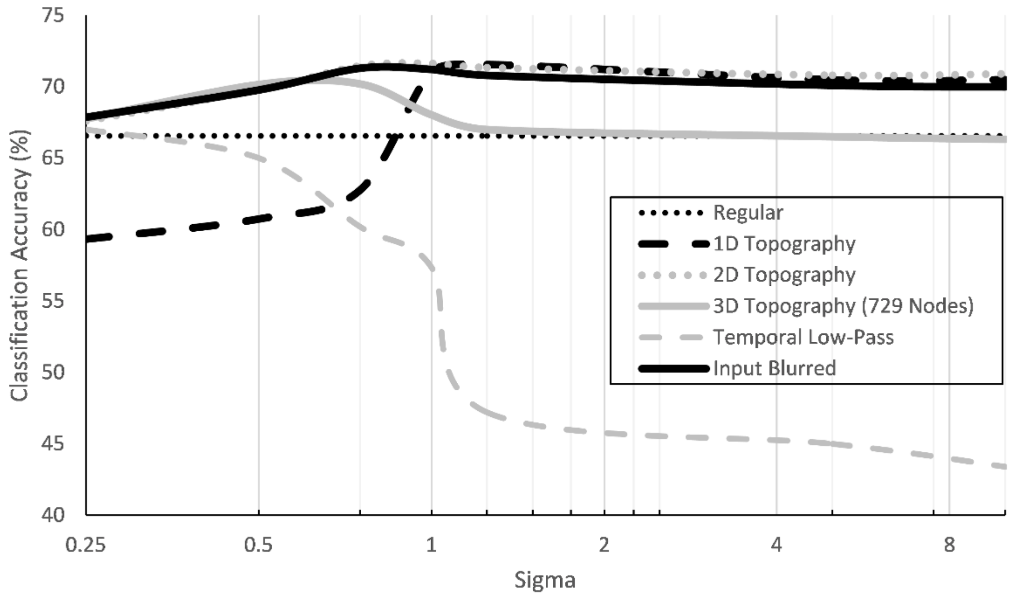


Figure 3.11. Classification accuracy on the CIFAR-10 test set vs sigma of the Gaussian function for each technique at 900 nodes.

It can be seen for the 2D and 3D topographies, as well as the input blurred method, that a larger sigma generally provides an increase in classification accuracy to some limit; as the Gaussian filter approaches a mean filter, the classification improvements level off. There is a peak performance with a sigma value in the 0.75 to 1.25 range typically, then the classification rate drops slightly but stays consistent as sigma increases. Note that peak performance is slightly less on the 3D topography, and this is due to the lesser number of hidden nodes used for comparison, given the cubic constraint mentioned above. The temporal low-pass method does not produce results much better than the regular RBM, and incurs a significant drop in performance as the sigma increases. Given that the qualitative analysis of the temporal RBM feature detectors are similar to the regular RBM, as discussed at the beginning of section 3.3, it seems that the temporal filter induces a weaker learning signal which produces similar but less developed features. Presumably, a temporal RBM's diluted learning signal means that it must be trained longer to achieve a similar performance to the regular RBM, without providing any real benefits.

To ensure that this performance increase is not a result of the test set and that the methods are in fact producing better visual feature detectors, the performances of the various biasing methods are tested on different datasets. Coates *et al.* show that a receptive field size of 6 provided the best results on CIFAR-10 classification, while increasing receptive field sizes gradually degraded performance [23]. To support the comparison, the best classification results for each biasing method trained on Hollywood2 and tested on CIFAR-10, using 6x6 receptive fields instead of the 8x8 image patches initially considered, is shown in Table 3.1. Table 3.2 shows the classification results when the same methods are tested on STL-10, also using a 6x6 receptive field, and with RBMs trained on the Hollywood 2 dataset.

Method (RBM)	**Classification Accuracy (%)**
Raw Pixels (reported in [23])	37.30
Regular RBM	71.40
Topographic 1D (sigma = 1.25)	73.77
Topographic 2D (sigma = 1.00)	73.90
Topographic 3D (sigma = 0.75)	72.52
Temporal low-pass (sigma = 0.30)	72.12
Input blurred (sigma = 1.50)	**74.33**

Table 3.1. Best classification results for each method, with 1600 hidden nodes, using 6x6 image patches from the CIFAR-10 dataset. Note that the topographic 3D RBM uses 1331 nodes such that all dimensions could be equal.

Method (RBM)	**Classification Accuracy (%)**
Raw Pixels (reported in [23])	31.80
Regular RBM	49.11
Topographic 1D (sigma = 1.25)	**52.98**
Topographic 2D (sigma = 1.00)	52.46
Topographic 3D (sigma = 0.75)	50.88
Temporal low-pass (sigma = 0.30)	49.29
Input blurred (sigma = 1.50)	51.51

Table 3.2. Best classification results for each method, with 1600 hidden nodes, using 6x6 image patches from the STL-10 dataset. Note that the topographic 3D RBM uses 1331 nodes such that all dimensions could be equal.

STL-10 contains much less labeled data than CIFAR-10, causing it to suffer from a lower classification rate. However, the values reported here are consistent with the performance of single-layer networks using this classification protocol on these datasets [23]. All of the methods perform measurably better than the regular RBM, with the exception of the temporal low-pass filtered RBM.

Furthermore, to ensure that the performance increase is not a result of the training set and patch selection method, and that the biasing methods are indeed producing better visual feature detectors, the performance of the various methods is also tested on different datasets after being trained on a static image dataset instead of the dynamic Hollywood2 dataset. The classification accuracies of each biasing method tested on various datasets, after being trained on CIFAR-10 at 1600 nodes, is shown in Table 3.3. The same sigma values as before are used as an indication of the best performing parameters.

Method (RBM)	**Classification Accuracy (%)**		
	CIFAR-10	***STL-10***	***COIL-100***
Regular	73.82	51.49	84.24
Topography 1D (σ=1.25)	75.41	52.21	85.69
Topography 2D (σ=1.00)	**76.24**	54.06	86.28
Topography 3D (σ=0.75)	75.65	**54.76**	84.28
Input Blurring (σ=1.50)	74.76	52.10	**86.82**

Table 3.3. Classification results, with 1600 nodes, when tested on CIFAR-10, STL-10, and COIL-100, after being trained on CIFAR-10. Uses the sigma values which produce the best results on CIFAR-10 for each method. Note that the topographic 3D RBM uses 1331 nodes.

Again, each of the considered biasing techniques produces a feature representation that gives a notable increase in performance over the regular RBM across all datasets; some greater than others depending on the dataset. With such a large amount of features, 1600, representational power tends to saturate and classification accuracy plateaus. Therefore, it is interesting that the techniques still produce a fixed increase in classification performance despite hidden layer size.

To evaluate how well the representation learning generalizes to different data, Table 3.4 shows the results of training and testing on the same dataset (using different parts of it) for STL-10, COIL-100, and MNIST.

Method	**Classification Accuracy (%)**		
	STL-10	***COIL-100***	***MNIST***
Regular	51.55	**78.74**	97.57
Topography 1D (σ=1.25)	53.22	75.66	**97.97**
Topography 2D (σ=1.00)	**54.34**	75.59	97.88
Topography 3D (σ=0.75)	54.31	76.06	97.80
Input Blurring (σ=1.50)	52.28	78.69	97.38

Table 3.4. Classification results, with 1600 nodes, when tested on STL-10, COIL-100, and MNIST after being trained on themselves. Uses the sigma values which produce the best results on CIFAR-10 for each method. Note that the topographic 3D RBM uses 1331 nodes.

It can be seen that the biasing techniques, when the RBMs are trained on STL-10, produce similar increases relative to the regular RBM as when they are trained on CIFAR-10, as shown in Table 3.3. STL-10 contains very similar images to CIFAR-10, including a lot of background imagery. These training samples are taken from real images, and as a result, they produce similar performance.

But training on a simpler dataset, such as COIL-100, causes the biasing to produce worse results than the regular RBM. COIL-100 contains a slightly different visual domain, as shown in Figure 3.5, where the object is isolated and does not contain background imagery. It also requires that the network learns the characteristics of the object from a very small amount of image data; from 4 separate angles. The dip in performance is likely due to the large amount of uninformative patches, since there is a lot of black space in COIL-100 due to a lack of background. The considered biasing techniques actually degrade performance from the regular RBM in this domain.

MNIST offers the most different visual domain, as shown in Figure 3.6, in that it is not real-world image data but handwritten characters. But the topographic biasing techniques do provide a notable increase in performance over the regular RBM. Given the improved results on MNIST, a dataset which is markedly different from the object recognition datasets, and where a 0.2% increase in accuracy is considered statistically significant [152], it is visible that the topography techniques can translate to another visual domain.

In general, the topographic methods perform best, and classification accuracy is boosted as the features benefit from the shared information within the neighborhood during training. At sigma values for the biasing Gaussian function with peak classification performance, the 1D topography produces similar results to the 2D and 3D topographies. However, it is computationally simpler, since its 1x3 kernel only requires 2 neighbors. Thus, sharing a small amount of information between nodes produces features that result in a statistically significant improvement in classification accuracy, and this improvement does not correlate with the number of neighbors, given that the results of the 3D topography are not much better than the others. For these reasons, the 1D topography appears to be a better solution than the other topographic methods when training time is important. Otherwise, they are comparable in terms of accuracy. The temporal low-pass biasing performs very poorly, and seems to only attenuate the effect of training; the results show degrading performance compared to the regular RBM whereas the other techniques improve performance overall. Finally, the basic input blurring method provides a surprising improvement in classification performance over the regular RBM across the real-world image datasets, all while learning quite different looking features.

3.3.2 Invariance

The invariance score, calculated by Equation (3.9), is used to evaluate how invariant the learned features are. For this score, the biased RBMs were trained on grayscale versions of Hollywood2, and the optimal stimulus for each hidden node was determined by searching through all artificially-generated possible black and white line segments, with all possible thicknesses, that fit in the image patch and selecting the one with the highest response. Since most features nodes respond to shapes similar to lines, as exemplified in Figure 3.7 and Figure 3.8, they were used as the shape by which the optimal stimulus is defined as described in section 3.2.6. An example optimal stimulus corresponding to a hidden node feature detector is shown in Figure 3.12. A set of non-optimal stimuli are also shown in that figure, where each stimulus is non-optimal since it does not match the position and orientation of the feature detector.

The most common invariances are to affine transforms, so a measurement of a feature's resistance to translations and rotations is evaluated as the transformations of interest. Since 8x8 patches are small enough that minor transformations will generate notable artifacts due to coordinate rounding, transformations are both performed by upsampling the patch containing the optimal stimulus to a higher resolution, performing the rotation or translation, and then downsampling back to 8x8. This should provide more realistic representations of real-world transformations to measure the network's performance.

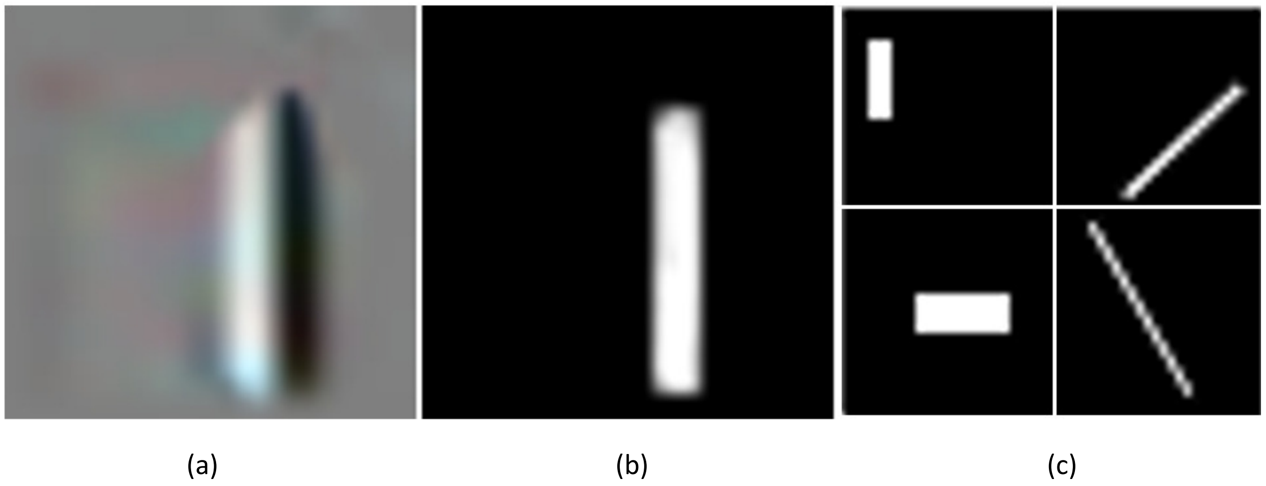


Figure 3.12. (a) Hidden node feature detector. (b) Optimal stimulus. (c) Examples of non-optimal stimuli that were tried before optimal stimulus was found. These are displayed at a higher resolution than 8x8 so they are easier to visualize.

There is an assumption that each technique performs similarly with grayscale data as it does with colour data. Networks operating on full colour data could have been used, along with real-world image patches as the stimuli, however the simplified method has some advantages. Primarily, the centroid of the line can be found, such that rotations correspond to rotations about the centroid as opposed to the center of the patch, making it a better analysis of rotation invariance. Also, there are fewer optimal stimuli per hidden node with the grayscale version and lines, meaning that there will be less variability in the results than if limited amounts of random image patches are used.

Translations of -4 to +4 pixels in both axes incremented by 1 pixel, and rotations of -45 to +45 degrees incremented by 1 degree, were used as local trajectories according to Equation (3.7), and the final scores, calculated by Equation (3.9), were summed together. As in [153], a firing threshold for each node is selected such that its firing rate when applied to random stimuli, G(j), is 0.01. The top 10% of the scores were used to calculate the means, which are reported in Table 3.5.

Biasing Method	**Invariance Score**
Regular RBM	22.7
Topographic 1D RBM (sigma = 1.25)	13.1
Topographic 2D RBM (sigma = 1.00)	34.4
Topographic 3D (sigma = 0.75)	48.6
Temporal low-pass filter (sigma = 0.30)	25.7
Input blurred (sigma = 1.50)	56.4

Table 3.5. Invariance scores of all biasing methods, with 225 nodes (216 for 3D topographic RBM) trained on grayscale versions of Hollywood2 and tested on a range of artificially generated translations and rotations.

As observed previously from the classification results, the topographic and input blurred RBMs outperform the regular RBM and temporal low-pass RBM. However, the invariance scores do not necessarily reflect that invariance is the main reason for this improvement. The topographic 2D and 3D

RBMs, as well as the input blurred RBM, have higher invariance scores and higher classification results, however the topographic 1D RBM has a lower invariance score than even the regular RBM, yet still produces higher classification results. The discrepancy in invariance scores between the 1D and 2D topographic RBMs is significant, but that does not translate into a discrepancy in classification accuracy. As explained in section 2.4.3, invariance is often credited with being a crucial step in achieving better classification rates. The results in this thesis show that a higher invariance in a network's features may contribute to better classification, but there are many more factors at work than just that.

3.3.3 Discussion

Overall, the 1D topography biasing produces comparable results to the 2D and 3D topographies. Though the features and topographic map are not as visually clean as the 2D and 3D biaising techniques, the 1D topography is computationally simpler, as it is a 1x3 kernel and only requires 2 neighbours in the convolution. The 2D topography requires a 3x3 kernel, meaning 8 neighbours, while the 3D topography requires a 3x3x3 kernel, resulting in 28 neighbours. Thus, sharing a small amount of information between nodes produces a significant improvement in feature detectors, and this improvement does not correlate with the amount of incorporated information. For these reasons, the 1D topography is typically a better solution than the other topographic biasing techniques. The computational complexity scales linearly with an increase in image size or kernel size in one axis. An increase in image size or kernel size in two axes or three axes scales with a power of two or three, respectively.

Using basic Gaussian smoothing as an input blur also produces a surprisingly notable increase in classification accuracy, enough to exceed the improvements shown by topographic methods at lower hidden node counts. Even at higher node counts, it exceeds the regular RBM by 2-3 percent depending on the dataset.

An interesting finding in the topography methods is that, despite the topographic arrangement, the classification accuracy is boosted even without pooling as in TICA or ISA networks. The features themselves benefit from the shared information within the neighbourhood. The arrangement itself is not important after training, though pooling would likely increase the results further, it would bleed over into becoming a two-layer network or at the very least using extra supplemental information coming from a more complex architecture; as a result, it is not discussed in this thesis where the focus is kept on single-layer networks in order to present a fair comparison between various methodologies.

As for an overall reason as to why these techniques produce better representations, there are various explanations. With input blurring, the edges are blurred, resulting in the network learning blurry edges as features that are inherently invariant to small transformations of a sharp feature. With the induction of topography by sharing information between neighbouring nodes, it makes sense that nearby features will respond to similar features such that the response of one of the neighbouring nodes will still respond strongly to a large transformation. In a sense, it also blurs the features. The softening of features makes them more flexible to firing under slightly different projections of that feature. However, the 1D topography showed an invariance score that was less than any of the techniques, yet it produced excellent classification results. Therefore, the ability to form a good representation goes significantly beyond the network's ability to generate invariant features.

3.4 Summary

This chapter reported on an experimental comparison between the biasing effects of Gaussian filters in unsupervised training of single-layer networks to evaluate their relative performance when it comes to learning invariant detectors for image features. In addition to improvements in classification results, an invariance metric was used to measure how invariant the features are to transformations of their optimal stimulus.

Using the RBM as a common framework to test the different training methodologies, it was possible to isolate the performances of each. Though it is possible that another framework, such as ICA, would produce different performances, it is more likely that methodologies to induce invariance would be similar across different frameworks.

In single-layer networks, the biasing to produce topography quite obviously produces better classification results than without biasing. Temporal low-pass filtering with video does not produce features that are any better at classification than the RBM without biasing, yet their invariance measure is higher. Blurring the input images produces a surprising increase in classification accuracy and invariance metrics, often outperforming the other biasing techniques. Also, though the 2D and 3D topographies produce features with higher invariance, the 1D topography provides classification improvement while the invariance metric shows a lower tolerance for feature transformation. It is also shown that 1D topography is just as good as 2D and 3D topography for classification, yet with a much lower computational complexity. The topographic method produces better features without exploiting known characteristics of visual data, such as temporal and spatial visual consistency. This makes it an interesting method to improve discriminative feature learning in other domains where temporal low-pass filtering and image blurring may not be applicable.

The comparison between the invariant properties and classification results of different learning methodologies for unsupervised networks produces tangible evidence that, despite differences in the approach, improved features can be achieved by biasing the network appropriately. Invariance is an important property to allow compact representation, and strikes a balance between generalization and specificity. For purposes such as object classification, learned features must be selective in some aspects and invariant in others, and improved invariance can often be correlated with better features. However, improved classification results, and the idea of learning good features should not necessarily be credited to invariant features since there are more factors involved, evidenced by the increased classification performance with the 1D topography despite a lower invariance score. This thesis shows how the effect of neighbouring information can be used in a simple manner to produce improved performance. The lack of research in quantifying and characterizing the effect of these biasing techniques and their performance under different conditions makes this a useful contribution in the field of unsupervised visual feature learning. Overall, this part of the thesis aims to shine some light on simple preprocessing and regularization methods that are best suited for balancing the properties of invariance and achieving good features. This research has been published in [154, 155].

3.5 Transition to Motion Tracking

Inherently, using a limited amount of hidden nodes to learn features from real-world data means each feature will be invariant to some degree in order to achieve the objective of minimizing the error. If the features were narrowly selective, a very large number of nodes would be required to sufficiently cover the necessary features to define objects. And then, the ability to learn patterns or objects would not be particularly robust as the features perform little generalization.

In comparison to achieving invariant features, learning motion features has very opposing requirements. Motion detection requires sensitivity to slight changes. It also needs generalization to different features but selectivity to their motion. Also, real-world video is a requirement instead of a complementary feature.

The recent move towards unsupervised learning, learning from large amounts of data, and focusing on self-organization using real-world images or video has provided excellent results in a number of domains likely due to the data-driven approach reducing the bias - and surpassing the insight - of human designers. In the context of these principles, achieving motion sensitivity has some similarity to achieving invariance. In order to contribute to that developing movement, a topic of interest is the ability of a network to self-organize based on real-world video input for the purpose of motion detection. An application suited to testing motion detection is motion tracking, which is one of the most researched problems in machine vision, yet one of the least common applications of biologically-inspired self-organization and unsupervised machine learning. Given the studied discriminative properties of the RBM in this chapter, their role in deep networks, its known generative properties, and the focus on using them to create action as well as perceive it, motion tracking is a suitable progression which will be further investigated in Chapter 4, leading to the principal contribution of this thesis.

Chapter 4 A Biologically-Inspired Self-Organizing Motion Tracking Model

In the previous chapter, the idea of using unsupervised learning to create self-organized feature detectors was covered. This chapter expands on the research on biologically-inspired self-organizing models by introducing a novel model to tackle the problem of motion tracking. Aside from the use of the restricted Boltzmann machine and the general philosophy of the work, the methods described in Chapter 3 are not built upon here, and only make a brief appearance in an evaluation of feature representations conducted in section 4.1.5.

The method proposed in this chapter is a novel framework which learns to execute motion tracking behavior through visual input and window control output. Though the result is motion tracking behaviour, the goal of the research presented in this chapter is to evaluate the integration of perception and action in one self-organizing system driven by real-world visual data. This is important, both for the purposes of expanding data-driven methodologies to cover representation and agency, as well as understanding how similar behaviour to those observed in the brain can emerge in a computational model.

Due to its biological inspiration, and given the strong ties of the biological theory to the computational application considered here, some terms will be used interchangeably depending on whether it is addressed from the computational perspective or the biological perspective. The region of interest corresponds to the receptive field, where a limited window of visual data is obtained from the retina or, in the computational case, video. Gaze will correspond to the position of the receptive field or the position of the region of interest. Gaze movement will correspond to receptive field movement or moving the region of interest. Pursuit and motion tracking will be used interchangeably. Saccades correspond to distinct movements of the gaze.

The idea of self-organization, covered in section 2.1, provides the basis of this thesis. The idea that a system can arrange itself from a random state to a functional state is a suitable starting point to show that complex behaviour can emerge through a basic learning rule. In brain modeling, this arises from the idea that the brain itself starts of not knowing much, except for some basic guidelines on how to learn. Considering that genetic information alone is unlikely to provide enough data for the brain to properly encode trillions of connections between neurons, it is thought that sensory data as well as genetic instructions over time govern the ultimate arrangement of the brain. Being a strongly-supported perspective in brain modeling, self-organization is an apt solution to model motion tracking behaviour guided by its biological equivalent in the brain: pursuit. Computationally, self-organization is facilitated by unsupervised learning. In some sense, in unsupervised learning, the learning procedure and its structure can be considered the genetic information, while the training data can be considered the sensory input. As a result, it makes sense to take a disordered system which is capable of learning, and ideally transform it into a system that has learned how to do the task at hand. The latter, here, is motion tracking to be learned using only real-world data. The usefulness of general-purpose unsupervised machine learning algorithms is that they begin with the capability of learning all possibilities, and result in becoming narrowed down to only encoding valuable possibilities. This encoding corresponds to an internal representation, and ultimately a method by which action can take place.

As stated in the introduction, this thesis is motivated by the concept of a fully self-organized system explaining behaviour that is commonly found in biological systems. In literature, there are many examples of representation emerging from self-organizing methods, however, there is less research on where action also emerges from self-organizing methods. The integration of action and representation through a single mechanism is an interesting concept to explore. Methods are commonly found where action and representation are separated, such as that of the traditional control system where sensory information is processed and some logic dictates what actions must be taken. These methods integrate modules of different functionality, and pass specialized information between them. This chapter will use a general-purpose generative unsupervised learning method to show that the desired behaviour can arise without a customized framework.

The generative model is required, since information must pass bi-directionally to both represent real-world data and reconstruct that same data if the same mechanism is to be used for training and execution. The work presented in this chapter represents control of the gaze by bi-directional nodes, which both propagate the movement of the receptive field upwards during training and control the receptive field position through top-down generation during execution, both of which can occur simultaneously.

Naturally, the execution of motion tracking system can produce gaze shifts corresponding to motion if it knows the association. However, the goal of this system is not just execution, but learning how to execute. This requires the system to learn the association between visual motion and gaze shifts through some unsupervised mechanism. Also, it is desirable that the same framework is used for training as well as execution, instead of somehow learning associations and then applying them in a different architecture. To the best of our knowledge, the capability of learning the association between visual motion perception and gaze control entirely from real-world videos using the difference in image contents as scalar reinforcement signal has not been achieved in literature.

Gaze movements can occur in any direction, motion can occur in any direction, yet these are separate concepts internally. Provided that the appropriate motivation can be found to link motion and congruent gaze movements, a simple mechanism is all that is needed to facilitate the emergence of more complex behaviour in the form of pursuit.

Retinal constancy is the simple mechanism required to create the association between motion and desired eye movements. This mechanism is the most important part of the proposed model and one of the major contributions of the research presented in this chapter. While finding the difference between images is not new, getting the brain to learn based on a retinal projection staying the same, and getting a model to learn based on receptive field contents staying the same, has not been extensively explored in the literature. The assumption made is that, given no prior understanding of motion information being carried from frame to frame, the most predictable event to occur is that there is no motion. This is a plausible assumption biologically-speaking, because the majority of visual exposure to the newborn is done within the context of a relatively static environment. Specifically, saying there is no motion is thought to mean that there is no motion relative to the retina; there is retinal constancy between frames. As the system's best guess is that there is no motion relative to the retina, its behaviour is to arrive at a state such that this guess is fulfilled. When its gaze is on a stationary object, it needs to do nothing to fulfill this guess. When its gaze is on a moving object, and given its agency over gaze control, it needs to move its gaze to match the speed and direction of the moving object to fulfill its guess. Thus, the gaze moving to match the speed and direction of the object will produce no motion relative to the retina. From the perspective of reinforcement learning, retinal constancy is to be rewarded. From the perspective of free-energy learning, the system must associate motion to gaze control such that retinal constancy is maximized, relating to a high predictability of the visual contents and minimal surprise.

By using gaze shifts, analogous to biological saccades, as a method to traverse the local visual space, retinal constancy can drive the network to learn correlations between naturally-occurring movements in video and gaze movements. The desired combinations will have a high retinal constancy. Most saccadic movements will result in low retinal constancy. Yet, the network will learn strongly from the rare saccades that move congruently with the motion in its receptive field. Thus, saccades are an important aspect to find the right associations for self-organization. Essentially, the network tries to learn situations where motion occurs, the gaze is moved, and the visual data remains mostly the same. Random traversal of the gaze shift input space combined with the variety of motion present in real-world video data eventually leads to finding the correct associations resulting in self-organization of motion tracking. The novelty here is that a single value representing similarity between receptive field snapshots, formulated as the retinal constancy, is able to self-organize the entire system to generate motion tracking behaviour.

A novel architecture is presented, which is an entirely unsupervised approach, including the use of a GRBM to learn transformations from real-world data, and ultimately its connection to an RBM which learns how to shift its receptive field such that change in that field is minimized between timesteps.

Using real-world video instead of synthetic data, the work presented in this chapter aims to show self-organization of a network to learn motion tracking based on a simple learning principle when driven by data alone. The data driving the learning of motion tracking behaviour is a single similarity value between frames of video fed to a single node, or as a coefficient of the learning rate, and is sufficient for the network to learn spatial correlations between gaze control and the associated motion detection features. The model is also tested by performing tracking on real-world data.

An RBM is used to implement both motion encoding and motion tracking for two reasons: its ability to learn in an unsupervised manner and its function as a generative model. Furthermore, its biological similarities make it an elegant solution for the purpose of showing that this behaviour can be learned through only the knowledge of local connections in intermediate stacked representations, thus forming a deep belief network.

This technique shows a generalized system’s ability to learn the same behavior as a specialized motion tracking architecture by using a simple motivation mechanism without supervision or hand-crafted design.

The method consists of three modules: The receptive field, the motion encoding module, and the motion tracking module. The receptive field receives the raw visual data, performs rudimentary preprocessing, and passes the output to the motion encoding module. The motion encoding module is a network which learns in an unsupervised manner to encode motion behaviour in the receptive field and passes the output to the motion tracking module. The motion tracking module decides if and how the gaze should move and what part of the image the focus should be on. The combination of these modules should drive a receptive field to track motion with the added novelty of being learned in a fully unsupervised manner. This chapter experimentally demonstrates the successful combination of the motion encoding module and the motion tracking module to produce motion tracking behaviour; the proposed model will be detailed in section 4.1, its performance will be discussed in section 4.2, and its comparison to existing methods and its biological parallels will be discussed in section 4.3.

4.1 Model Description

As covered in section 2.5, state-of-the-art motion tracking solutions in machine vision typically consists of several design concepts: finding an appropriate representation for the target, a method to detect the target and match it from frame to frame, constraints to reduce computation time and false positives, and filtering to compensate for error in position by taking past information into account. Biological counterparts to these concepts exist, but are less precisely defined and the mechanisms are not as well-understood.

The model proposed in this chapter introduces a biologically-inspired principle for motion tracking, which self-organizes through exposure to real-world data only. It offers novel perspectives on many properties of machine vision solutions to this problem, and leads to many of the design concepts which are typically handcrafted to naturally emerge through self-organization.

The relationships between the high-level modules for the motion tracking model are shown in Figure 4.1.

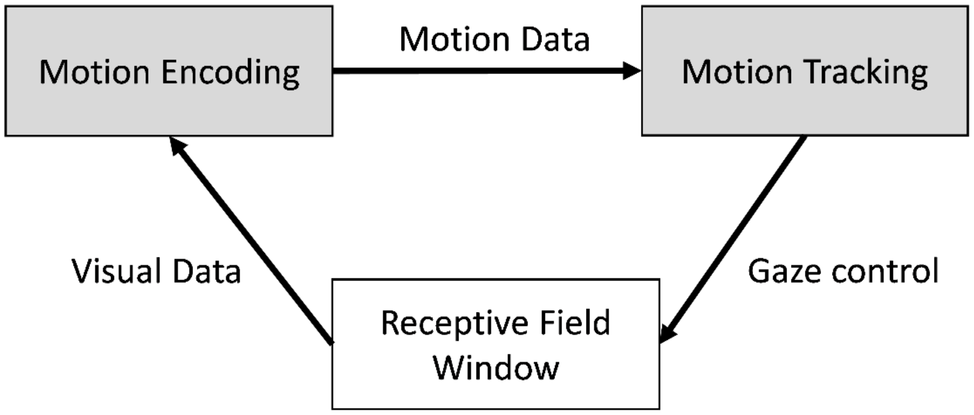


Figure 4.1. Proposed motion tracking model with the complex logical processing modules, motion encoding and motion tracking, highlighted.

This combination of modules is a trivial overview of a generalized motion tracking system, where the logical modules of motion encoding and motion tracking occupy the bulk of the processing and can be implemented in a variety of ways. From the machine vision perspective, motion encoding can consist of representing the target, estimating possible target matches from frame to frame, within constraints. In the same perspective, motion tracking can be considered as selecting the best possible target match, considering that as the true motion, and moving the region-of-interest to reasonably match that target such that it can be used for future calculations.

The novelty in the proposed work is that the logical modules are all self-organizing through exposure to visual data. The motion encoding module learns from visual data, while the motion tracking module learns from the output of the motion encoding module. The motion encoding module learns to represent motion from visual data and the motion tracking module learns gaze control based on the encoded motion. Each of these modules is implemented as a network, effectively making the system become a two-layer deep network. Upon training of this deep network, it performs functions similar to its machine vision motion tracking counterparts.

The RBM is used as a mechanism by which these modules are implemented. It is an ideal mechanism to meet the goals of the proposed system. It is capable of self-organization via unsupervised learning, it has generative properties which can create action, and can be stacked such that one layer can be learned on the outputs of the other layer to form a deep network. The model is implemented with the structure shown in Figure 4.2.

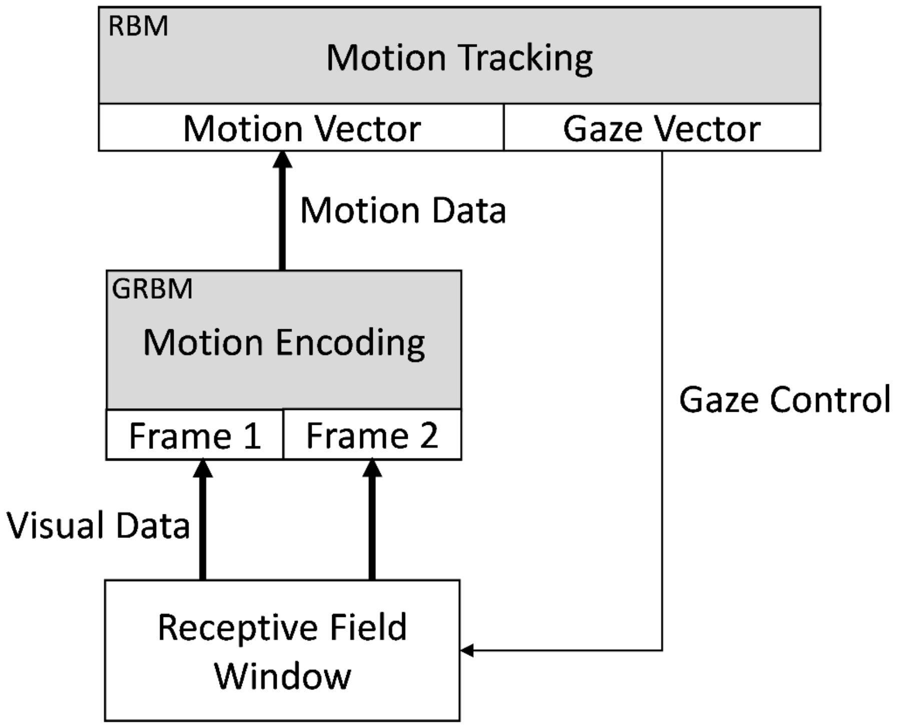


Figure 4.2. Modules forming the proposed motion tracking model arranged as layers in a deep network, with the mechanism used for that layer and its inputs, outputs, and their functions labeled.

Generally speaking, the network is a 2-layer deep network, with each layer performing a different task. The deep network contains a motion encoding layer which produces a representation of the estimated motion in the receptive field based off of two snapshots, and a tracking layer to learn associations between motion vectors and receptive field movements. The model contains a receptive field which is controlled by the gaze movement vector that is produced by the motion tracking layer.

The receptive field and motion encoding layer propagate data upwards, while the motion tracking layer behaves as a generative model and produces gaze movements as feedback downwards. The motion encoding layer, whose generated motion vector is concatenated with the gaze movement vector, is fully-connected to the motion tracking layer. The retinal constancy value is introduced to the motion tracking layer as a learning rate adjustment.

The motion encoding module can be considered a set of motion detectors which are learned in an unsupervised manner, implemented as a GRBM. The GRBM is used for its discriminative ability for encoding transformations and its ability to learn transformations from data via higher-order correlations. Exposed to real-world video, it comes to learn how to encode the speed and direction of the patterns presented to it in sequential frames. Following this training, the motion tracking layer is also trained in an unsupervised manner and implemented as a standard RBM. The motion tracking RBM learns to associate the encoded speed and direction of a target to the receptive field shift required to follow it. An RBM is used for this layer due to its generative properties and its ability to operate as autoassociative memory; it behaves similarly to Hinton's MNIST label generation [101], discussed in section 2.3.2.1, where outputs from a lower-level drive the generation of label data. In the current work, gaze movements are generated in a manner similar to those labels.

A more detailed view of the model, showing how the receptive field receives input from the video, how the retinal constancy modulates the motion tracking RBM, and the difference between execution and training with the gaze movement vector, is shown in Figure 4.3.

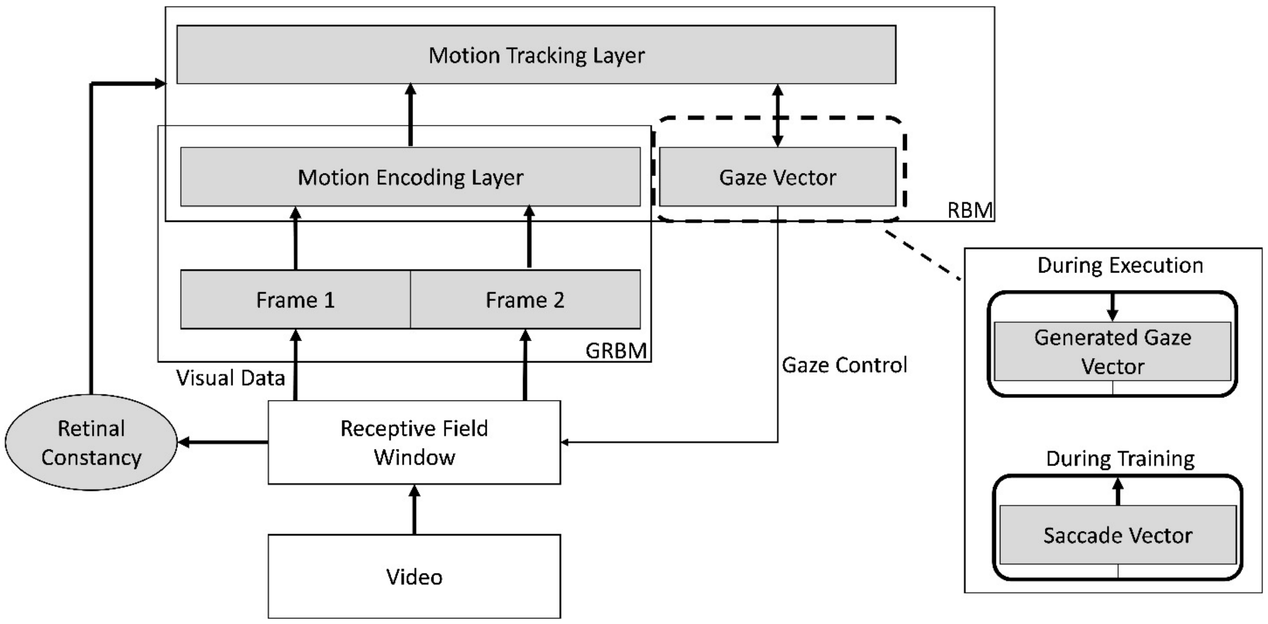


Figure 4.3. Complete system description, where the gaze movement vector is generated by the motion tracking layer during execution, and created randomly during training. The hidden activations of the motion encoding layer and the gaze movement vector are concatenated to form the input vector of the motion tracking RBM.

An additional property of the RBMs in this system are that they allow the stacking of these networks after training them individually. As explained in section 2.3.2, each layer can be trained on its own using the outputs from the layer beneath it. This process, known as greedy layer-wise training, is used to train the motion encoding GRBM first, then the motion tracking layer is trained based on the outputs of the GRBM. In theory, additional layers can be used on top of the GRBM to shrink the motion vector, and perhaps increase discriminative capabilities, prior to the motion tracking layer, to make the training simpler. Also, additional layers can be built upon the tracking layer to learn more complex patterns arising from the association of motion and receptive field movements, thus biasing the motion tracking through generative capabilities. However, these theoretical improvements remain beyond the scope of the work presented in this thesis, and two-layer layer-wise training is used.

The network has some general parameters as follows:

$$gaze \in \Gamma \tag{4.1}$$

$$h_{motion} = f_{motion}(x^{(t)}, x^{(t+\Delta t_m)}) \tag{4.2}$$

where gaze controls the receptive field and belongs to the set of all allowable transformations, Γ. h_{motion} is the encoded motion vector, and f_{motion} is the motion encoding output function, where $x^{(t)}$ is the receptive field image at time *t,* and Δt_m is the sampling delay between frames to detect motion. In the implementation, and in biology, the set of allowable transformations, Γ, is limited to only translations as that is all the receptive field and the biological eye can do (excluding more complex control such as focus).

There are slight differences in the formalization between training and execution. During training, it is as follows:

$$h_{tracking}{}^{training} = f_{tracking}\left(h_{motion}, gaze_{training}, r\left(x^{(t)}, x^{(t+\Delta t_m+\Delta t_g)}\right)\right) \tag{4.3}$$

$$gaze_{training} = \phi(\Gamma) \tag{4.4}$$

where $f_{tracking}$ is the motion tracking output function, based on the motion encoding output, h_{motion}, a stochastically generated gaze, $gaze_{training}$, and the retinal constancy value, r. Δt_g is the delay between the time the first frame is perceived to the time that the gaze catches up to the target. ϕ is the saccade generation function, which generates a random or pseudo-random gaze. The saccade generation will be covered in more detail in section 4.1.2.

During execution, the architecture remains the same, but the motion tracking output function relies solely on the motion vector. The gaze is calculated by $f_{tracking}'$, which is an inversion of the motion tracking output function that behaves generatively. This behaves as one Gibbs step for reconstruction. This is formalized as follows:

$$h_{tracking}{}^{execution} = f_{tracking}(h_1) \tag{4.5}$$

$$gaze_{execution} = f_{tracking}{}'(h_2) \tag{4.6}$$

Note that some liberties were taken with the convention. Only relevant information is included. For example, the input of $f_{tracking}$ will always have a fixed vector size, meaning that gaze and the motion vector will both be generated by $f_{tracking}'$ when implemented, however the only used value is gaze, so only gaze is listed. In addition, $f_{tracking}$ requires the same vector size during execution as training, however, only h_{motion} is useful for the calculation so it is the only one listed. Since there is no way to have an ignored input value during execution, practically, the values that are not used are set to some small non-zero value close to their average expected state such that they do not contribute too greatly to the calculations.

4.1.1 Receptive Field and Preprocessing

The receptive field represents the raw visual input of a receptive field and is moved by gaze control messages. Biologically, gaze control moves the eye. Computationally, it changes the position of the receptive field in the visual data source as shown in Figure 4.4. In the implementation, the visual data source is a video, and the receptive field is a small square region of interest in that video. The position in the full image is controlled by the gaze movement vector, which will be discussed later.

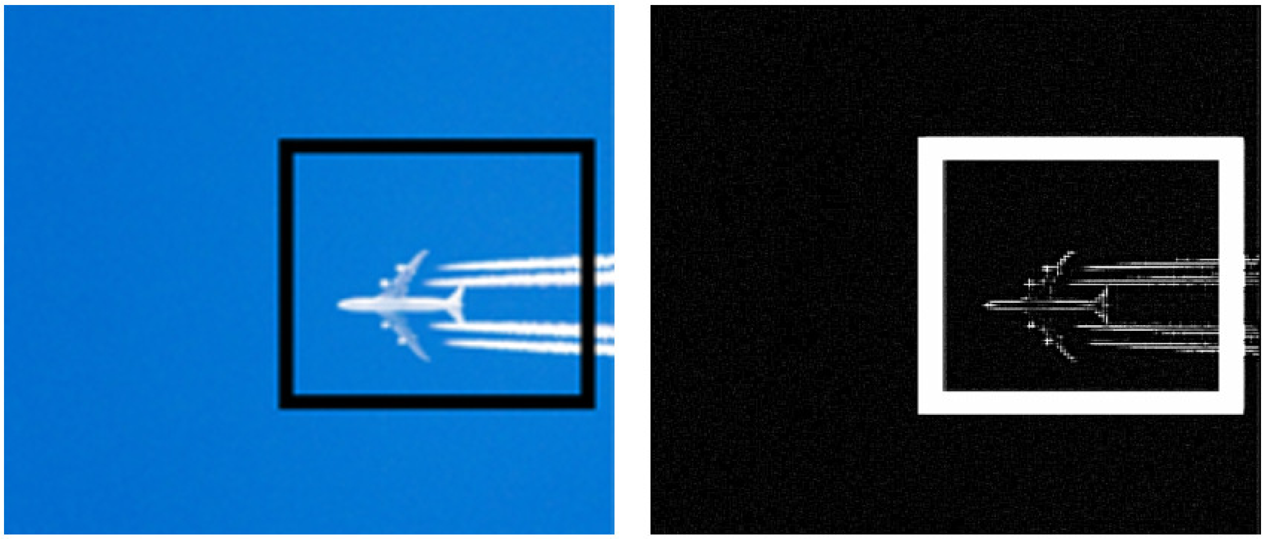

(a) (b)

Figure 4.4. (a) Receptive field in an image, and (b) image after grayscale conversion and redundancy removal via a center-surround mechanism.

Biologically, before the data is propagated, some preprocessing is done in the form of redundancy removal by center-surround mechanisms. In real-world visual data, adjacent areas are strongly correlated, and the most valuable parts of an image are those that break that correlation; typically those are changes in contrast or edges, as shown in Figure 4.4b. Computationally, an antagonistic center-surround calculation is applied to the grayscale input, and the normalized output is used as the receptive field data. Instead of using the difference-of-Gaussians calculation, a similar, but simpler, edge detection kernel is used as shown in Equation (4.7). Any standard edge detection procedure is likely to work, but this particular algorithm was the simplest to implement at the time.

$$c_{i,j} = \sum_{x=-n}^{n} \sum_{y=-n}^{n} \begin{cases} if\ (x,y) = (0,0) & ((2n+1)^2 - 1)p_{i+x,j+y} \\ otherwise & -p_{i+x,j+y} \end{cases} \tag{4.7}$$

The neighbourhood, *n*, describes the size of the surrounding area to be used in contrast to the center pixel, and $p_{i,j}$ is the grayscale pixel value at (i, j). This behaves as a local contrast detector, and helps reduce the amount of redundant information, resulting in the preservation of only important shape information, such as edges and contours.

If *n=1*, the operation behaves as a Laplacian filter of $\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$. So this can be considered a modification of the Laplacian filter.

The output is contrast normalized, keeping only positive contrast, by Equation (4.8) to ensure the data falls within the range of 0 to 1, where $c_{i,j}$ is one pixel of the edge detected image.

$$c_{i,j} = \frac{\min(0, c_{i,j})}{\max(c)} \tag{4.8}$$

4.1.2 Saccade Generation

An important part of the proposed theory is that saccades are used to traverse the local visual space during training. Instead of using a formal saccadic model, a simplified representation which generates gaze movements stochastically is used. Two methods are introduced, both of which produce good results, but each requiring a slightly different framing of the model.

4.1.2.1 Random Saccade

The most basic saccadic technique is randomly picking a gaze from Γ, defined in Equation (4.1), where each possible movement has equal probability. This keeps the saccadic model independent of the motion tracking behaviour, where motion tracking does not have any control over gaze control during the learning phase.

4.1.2.2 Motion-Driven Gaze

The second saccadic model is having the network generate a movement stochastically based on its inputs. Just as it behaves in the execution phase, the network generates a gaze movement in the training phase. However, instead of selecting the highest value in each axis, the value is randomly chosen according to a weighted distribution generated by the softmax function, described in Equation (4.9), of the motion tracking output vector.

$$P(y = j|x) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \tag{4.9}$$

where $P(y = j|x)$ is the probability that the j^{th} element in x is chosen as the gaze, x is the binary gaze movement vector produced by the motion tracking layer after being executed on the motion encoding outputs, and K is the vector size.

Using this saccadic technique simplifies the model into the same behaviour during training and execution such that it can learn and execute simultaneously, similar to biology. This implies that motion influences gaze movements before motion tracking has been learned. The results will show that both techniques produce similarly high results with similar training time; the only difference lies in that one model is more in-tune with a biological perspective while the other is a self-contained self-organizing model.

4.1.3 Motion Encoding Layer

The motion encoding module calculates motion information based on temporal visual data. Computationally, temporal visual data is represented by two image frames separated by a finite time interval, Δt_m. This can be considered similar to the delays used in the Reichardt detector, or the sampling frequency in other motion sensing techniques.

The motion encoding layer takes these two frames from the receptive field and produces a representation of the estimated motion between them, formally written as: $h_{motion} = f_{motion}(x^{(t)}, x^{(t+\Delta t_m)})$.

This encoding may be trivial when hardcoded, such as traditional machine vision methods, but a primary directive of this thesis is to allow such properties to be learned. As a result, the motion encoding layer must both learn the motion between frames and act as a motion detector, where its output encodes the the direction and speed of the movement in the receptive field.

The implementation uses the GRBM to learn transformations in visual data from frame to frame. These transformations effectively become an encoding of motion. Typically, the hidden layer nodes represent translations and skews when trained on real-world video data [81], thus making it an ideal mechanism for this task. Also, to meet the other goals of this thesis, it is a general machine learning algorithm, and is an extension of the RBM.

During execution, when the GRBM is presented with sequential frames as the visible vectors, the hidden layer produces a vector which describes the transformation, which can be considered the motion. This makes the GRBM a self-organized set of Reichardt detectors in a sense [61]. This idea is shown in Figure 4.5. In the figure, *x* and *y* are the visible nodes and represent images at time *t* and *t+ Δt*$_m$, respectively. These visible nodes are arranged as 1-dimensional vectors as the network has no prior knowledge of spatial relationships within the input data. The hidden nodes, *h*, represent the motion vector which is propagated to the next layer, while *f* are the factors.

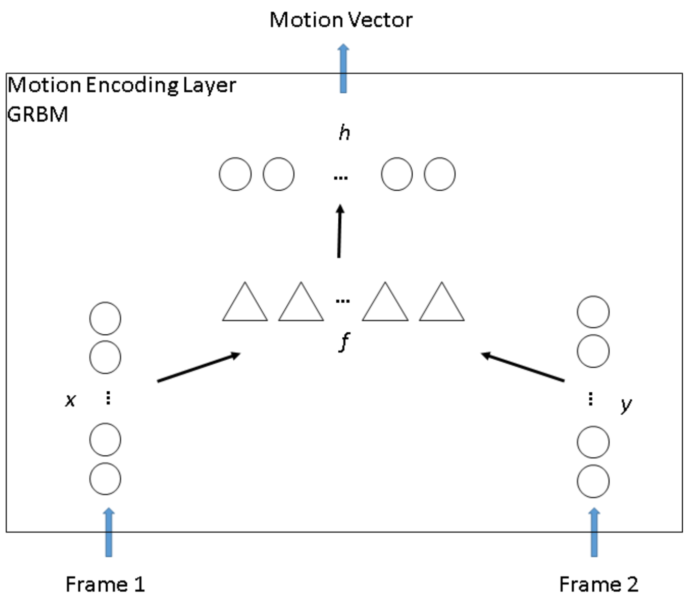


Figure 4.5. GRBM used for the motion detection layer.

It is important to note that the Gated RBM without factors was initially considered because of its simpler architecture and fewer choices of parameters. However, the cubic increase in parameters relative to an increase in image size makes it a less practical candidate than using the factors provided by the Gated Factored RBM, even if it is truer to the philosophy of this thesis.

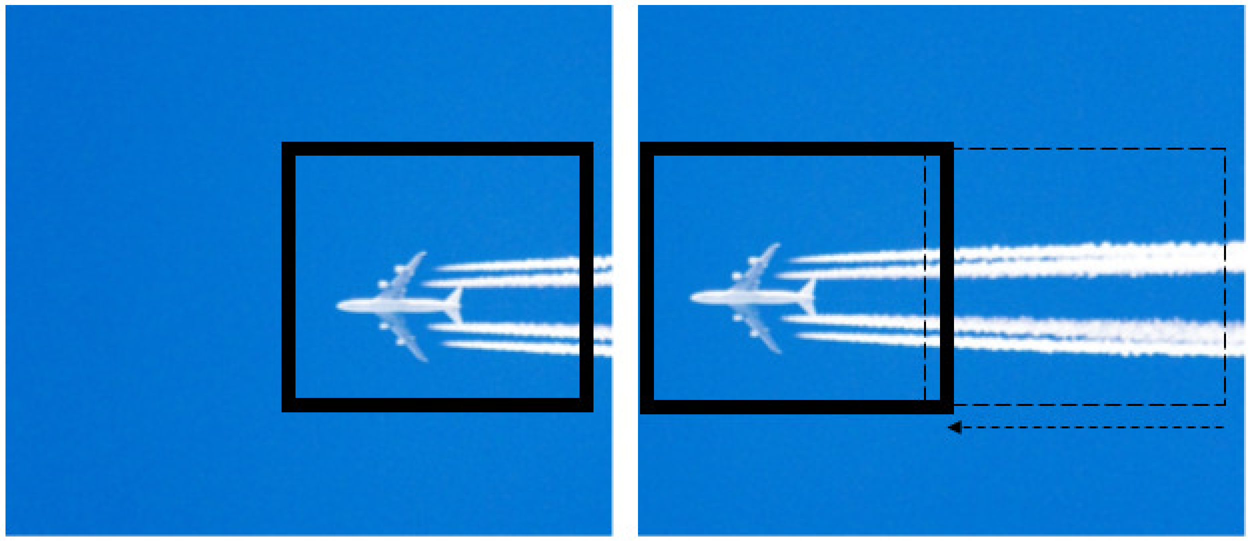
4.1.4 Gaze Movement Vector

Though the motion encoding layer will learn rotation, skews, scaling, and translation from real-world video, the action of tracking is simplified into only translations. While the motion encoding layer can be considered as an encoder of speed and direction, the gaze movement vector encodes the shift of the receptive field. In this model, it is assumed that the gaze can only translate the receptive field horizontally, vertically, and combinations of the two. This is sufficiently close to the biological eye which rotates about two axes to change its viewpoint in the scene. Thus, gaze movements are represented by a vector which encodes translation, as in Equation (4.10).

$$\Gamma = \{(r,s)|(r,s) \in \ ([-i,+i],[-j,+j])\} \tag{4.10}$$

where *(r,s)* is a translation vector within *[–i, +i]* horizontal range and *[–j, +j]* vertical range, and gaze $\in \ \Gamma$.

In the case of tracking in video, the gaze movement vector controls the position of the receptive field in the image. Based on the example presented above, Figure 4.6 shows the receptive field at time *t* and at t+Δt_m+Δt_g.



(a) (b)

Figure 4.6. Receptive field movement: (a) initial position of receptive field, and (b) position of receptive field after movement. Solid black outline is the current position, dotted outline is the previous position, and the dotted arrow indicates the latest movement.

Given that this vector is a part of the input layer in the Motion Tracking RBM, the connections between it and the Motion Tracking RBM hidden layer are bidirectional. During execution, the gaze movement vector is produced by the hidden layer based on the activations generated from the motion encoding vector inputs. It then controls the position of the receptive field accordingly. During training of the motion tracking RBM, the gaze movement vector is produced by saccade generation, explained in 0. As in the execution, gaze movement vector controls the receptive field, but for training, it also propagates the vector to the tracking module so it can learn associations.

There are three different representations that can be used to encode the gaze movement, each with benefits and detriments. They will be described in the following sections and ultimately compared in section 4.2.2.

4.1.4.1 Full Position Binary Vector

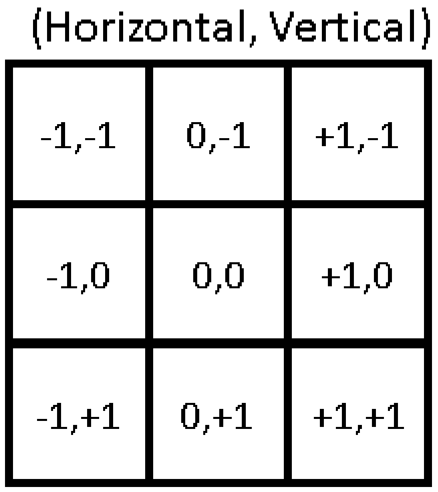
The full position binary vector representation is where the gaze movement vector describes the shift of the receptive field position which is mapped as a bit for each possible position. The vector size is given by: gaze movement vector size = (horizontal_range)*(vertical_range). When selecting a gaze based on this vector, the highest value in the gaze movement vector is chosen as the gaze movement, described in Equation (4.11).

$$(gaze_x, gaze_y) = argmax(gaze_{2D}) \tag{4.11}$$

where gaze$_{2D}$ is the 1-dimensional gaze movement vector produced by the motion tracking layer remapped into a 2-dimensional vector, and gaze$_x$ and gaze$_y$ are the selected discrete gaze movement coordinates. When creating this vector from a gaze to be passed to the motion tracking layer, it is a one-hot vector where the position corresponding to the horizontal and vertical shift of the gaze is set to 1 while the others are set to 0, as described in Equation (4.12).

$$gaze_{2D}(i,j) = \begin{cases} (i,j) = (gaze_x, gaze_y) & 1 \\ otherwise & 0 \end{cases} \tag{4.12}$$

For example, based on the topology depicted in Figure 4.7, a 2D mapped gaze movement vector of [1 0 0, 0 0 0, 0 0 0] indicates a -1, -1 pixel shift.



(Horizontal, Vertical)

-1,-1	0,-1	+1,-1
-1,0	0,0	+1,0
-1,+1	0,+1	+1,+1

Figure 4.7. Example of full position gaze movement vector description with horizontal and vertical ranges of -1 to +1 pixels.

This is the most rudimentary representation, and the network representation has no understanding of horizontal or vertical motion. Because of this, it cannot leverage training data from nearby samples to learn a better representation faster. However, it also does not suffer from confusion, meaning that each bit is only active on one gaze movement and does not represent multiple positions. The most negative aspect of this representation is that it increases exponentially with the gaze range, as it must represent each possible position.

Also, an increase in range causes a nonlinear increase in training time. The number of possible two-dimensional movements are equal to the square of the range; if the range is 5 (-2 to +2 pixels, 0 included) in each direction, the total number of possible movements in two dimensions is $5^2 = 25$. Only one of those possible movements is the correct one. Thus, the probability of the movement randomly corresponding to the eye movement is given by Equation (4.13).

$$P((r,s)_{random}=(r,s)_{actual}) = \frac{1}{(range)^2} \tag{4.13}$$

As the range increases, the number of possible shifts also increases, resulting in an exponential decrease in the probability of the preferred shift. This increases training time significantly, as the range increases since the number of useful vectors are reduced in probability. This provides additional justification for the use of alternate representations.

4.1.4.2 Axis-Split Binary Vector

The axis-split binary gaze movement vector describes the horizontal shift and vertical shift of the receptive field position individually. The vector is actually a concatenation of two one-hot vectors, implemented by dividing it into a half for left-right movement and a half for up-down movement. Each element represents a shift in that particular direction. The vector size is given by: *gaze movement vector size = (horizontal_range) + (vertical_range)*. When selecting a 2-dimensional gaze movement from the 1-dimensional gaze movement vector produced by the motion tracking layer, the gaze movement vector is broken up into the horizontal sub-vector and vertical sub-vector, and the gaze is selected from those, as shown in Equation (4.14).

$$(gaze_x, gaze_y) = (argmax(gaze_{horizontal}), argmax(gaze_{vertical})) \tag{4.14}$$

When creating a gaze movement vector to be passed to the motion tracking layer, two one-hot vectors are produced, one for each axis, as shown in Equation (4.15). The vectors are concatenated to form the final gaze movement vector.

$$gaze_{horizontal}(i) = \begin{cases} i = gaze_x & 1 \\ otherwise & 0 \end{cases} \tag{4.15}$$

$$gaze_{vertical}(j) = \begin{cases} j = gaze_y & 1 \\ otherwise & 0 \end{cases}$$

An example with the range of -2 to +2 pixels in both directions, resulting in a vector size of 10, is shown in Figure 4.8. In that case, a vector of 1000000010 would indicate a -2 horizontal pixel shift and a +1 vertical pixel shift, or a shift that is 2 pixels to the left and 1 pixel down corresponding to the standard image coordinate system where the top left corner is (0, 0). Notably, no more than one element in the horizontal portion can be set to 1, and no more than one element in the vertical portion can be set to 1. During execution, the position within each range with the highest value is chosen as the winning discrete shift in that direction.

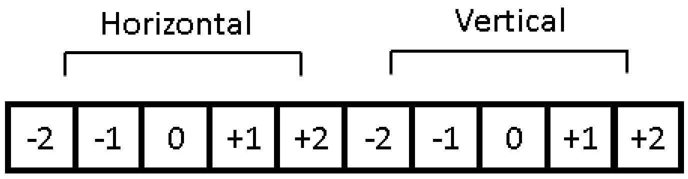


Figure 4.8. Example of axis-split gaze movement vector description with horizontal and vertical ranges of -2 to +2 pixels.

A single bit could be active for any position along its axis, meaning it has to encode a *range* number of different values. This can be a benefit, in that it is able to leverage similar transformations as more training data in order to train faster and more robustly. It is a detriment in that its discriminative power is reduced, as it can make more mistakes when it is responsible for many possible gaze movements.

4.1.4.3 Analog Vector

To provide a more realistic representation of control, we evaluate the use of analog vectors describing motion instead of the above-described binary vectors. Binary vectors require a node, or neuron, for each of many positions in the gaze movement. However, control of an eye would be better represented by one neuron for each axis, each able to drive the eye to the appropriate location in the corresponding axis, as opposed to encoding each possible position of the eye in each axis. Thus, an analog vector with only two values is proposed; one for each axis. The gaze movement can be recovered from the gaze movement vector by multiplying the singular gaze value from each axis by the range of that axis, as shown in Equation (4.16).

$$gaze_x = horizontal\_range \cdot gaze_{horizontal} \qquad (4.16)$$

$$gaze_y = vertical\_range \cdot gaze_{vertical}$$

The analog gaze movement vector is formed by scaling the horizontal and vertical components of the gaze to values between 0 and 1. This is performed by dividing the respective gaze component by the maximum range in those axes, and concatenating them into a single vector, as shown in Equation (4.17).

$$(gaze_{horizontal},\ gaze_{vertical}) = \left(\frac{gaze_x}{horizontal\_range}, \frac{gaze_y}{vertical\_range}\right) \qquad (4.17)$$

Figure 4.9 visualizes this mapping of a discrete value to a continuous value for the analog gaze movement vector.

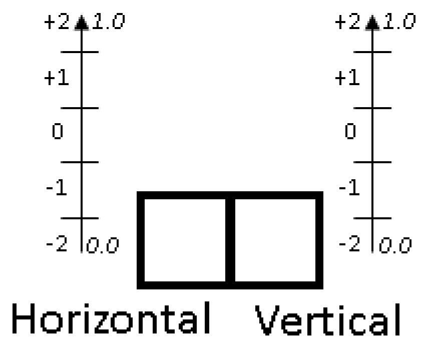


Figure 4.9. Example of analog gaze movement vector description with horizontal and vertical ranges of -2 to +2 pixels mapped to a value between 0 and 1.

This is a better encoding that allows the gaze to move to any position in the visual field and can represent more granular movement while keeping the network easily scalable. Using a binary vector requires that the number of input nodes to the motion tracking layer must also increase. Biologically, it is unlikely that there is a different neuron for each possible movement in each axis as represented by the binary vector. Ideally, an analog vector would only need one node for each axis, making it a more compact representation and a more realistic depiction of gaze control.

Another benefit is better learning of underrepresented patterns in the training data. Consider two points. The first is, given that the motion encoding may have a gradual change in activations with slightly varying speeds, it is possible that the effect of a pattern at one speed can affect the learning of a pattern at another speed due to the strong overlap in the encoding vectors. The second is that increasing ranges of gaze movement mean that there is a decreasing probability of random saccades producing the preferred movement to associate to motion encoding. Thus, all patterns are likely to be underrepresented due to limited training samples. Given the two aforementioned points, the benefit of analog nodes is that due to the overlaps in motion encoding vector patterns, analog gaze control can most benefit from interpolation of nearby training samples, more so than the full position and axis-split binary representations.

A comparison between full position binary vector representations, axis-split binary vector representations, and analog vector representations for the movement vector will be detailed in section 4.2.2.

4.1.5 Retinal Constancy

Retinal constancy is the primary concept that allows self-organization to occur such that motion tracking behaviour is attained, and is the most important contribution for both the proposed model as well as the theory of biological self-organized motion tracking behaviour. As stated earlier, the assumption made is that the most predictable scenario is that the image on the retina stays the same between timesteps, especially if there is no gaze movement. This is a reasonable assumption because neighbouring visual information, both spatially and temporally, is often the same. Retinal constancy is a value which defines how much the image on the retina has stayed the same.

When gaze movements occur, the tendency is that the image contents of the receptive field are not the same from frame to frame, thus retinal constancy is low. However, there are rare cases that an object in the receptive field also moves in the same direction as the gaze, notably keeping the receptive field image contents the same between samples, thus the retinal constancy is high. Aside from textureless areas, which are unlikely to be targets for motion tracking in the first place, it is most likely that the retinal constancy is high only when the gaze moves to match the direction and speed of the image contents. The network attempts to capture this notable occurrence and learn to associate motion of a certain velocity to gaze movements of a certain velocity. In a sense, the network is rewarded when the visual contents do not change much from frame to frame. Given that the architecture allows gaze movements to be generated, and allows visual information to be processed, it is suitable to use retinal constancy to associate gaze movements to changing visual information.

In this model, retinal constancy is the only way that desirable saccadic movements can be linked to detected motion. This concept is a cornerstone of the proposed model and of this thesis. It is a non-trivial solution, because motion relates to action solely through comparing snapshots of the receptive field. The comparison produces a singular value and does not describe how the visual contents change, and occurs despite temporal delays. The network does not have any predetermined knowledge of relationships between the receptive field and the action, yet learns to associate motion to the appropriate gaze control for motion tracking.

This comparison of receptive field snapshots is implemented as a comparison of two image frames from a video, and producing a single similarity value. Though a comparison between two frames is not a new concept in motion detection, using it to produce a single value that self-organizes multi-dimensional information is a novel contribution in the context of this original motion tracking learning framework and is the primary contribution of this thesis. The timing and overall concept of retinal constancy is explained in Figure 4.10, where an object moves in the receptive field over a time period of Δt_m, the receptive field moves over a time period of Δt_g, is evaluated by comparing the initial receptive field at time *t* to the current receptive field. The retinal constancy value is the similarity between the two snapshots of the receptive field. If the object produces a similar pattern on the receptive field, then retinal constancy is strong, and it is likely that the receptive field has moved with the target. Otherwise, it is weak, and the chosen movement is undesirable.

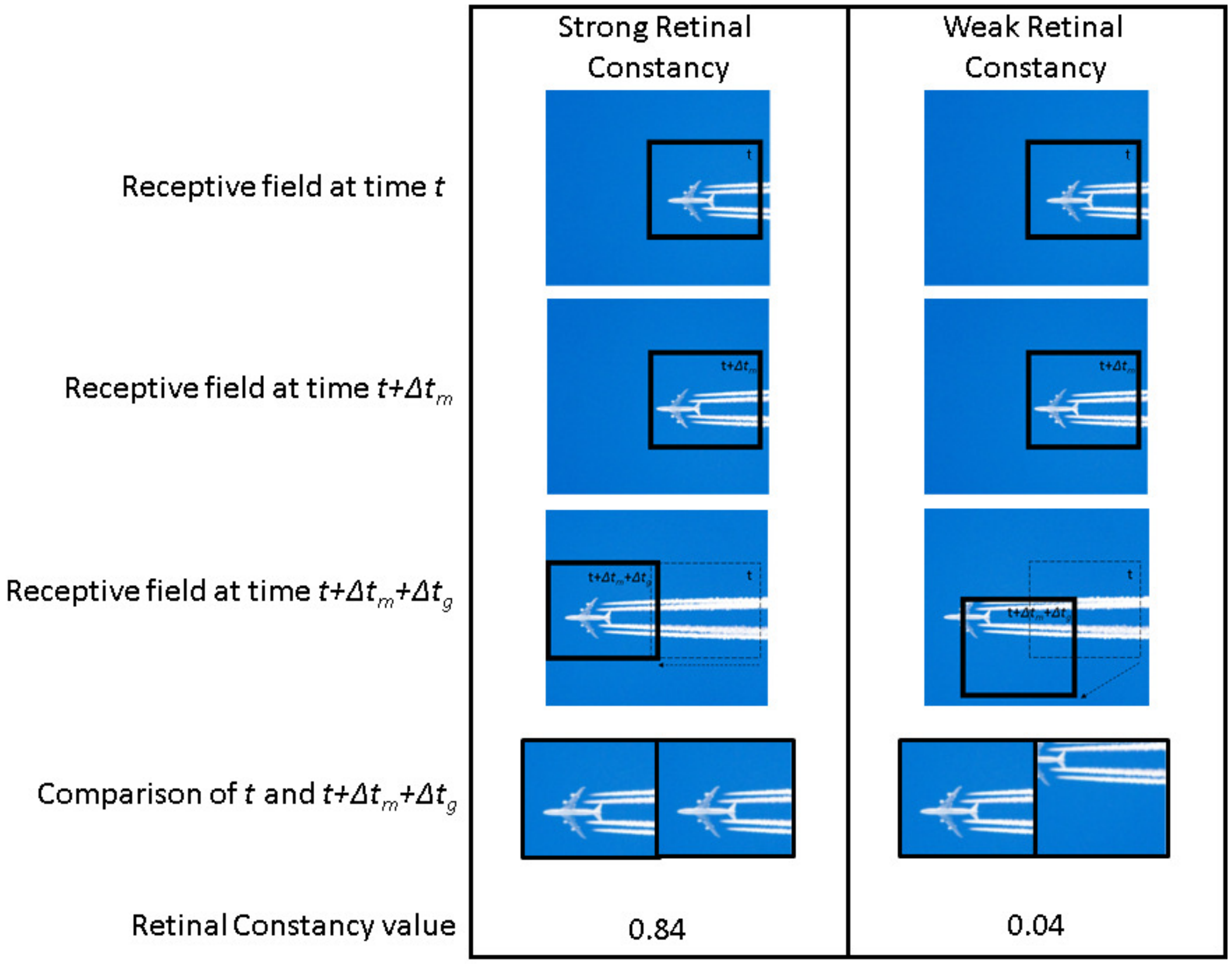

Strong Retinal Constancy
Weak Retinal Constancy
Receptive field at time *t*
Receptive field at time *t+Δt_m*
Receptive field at time *t+Δt_m+Δt_g*
Comparison of *t* and *t+Δt_m+Δt_g*
Retinal Constancy value
0.84
0.04
t
t+Δt_m
t+Δt_m+Δt_g

Figure 4.10. Receptive fields at t, t+Δt_m, t+Δt_m+Δt_g, and the corresponding retinal constancy.

The use of retinal constancy to have the motion tracking RBM learn favourable patterns can be considered a biasing of autoassociative memory. In a sense, retinal constancy assigns a confidence that the pattern is valuable such that the RBM throttles how much its internal representation changes to accommodate it. The biasing is performed by weighting some patterns more than others to give priority during learning. Most applications make use of all training data equally, assuming that the dataset is wholly valuable. In some applications, this may not be the case, and only a subset of the training data will be the desired distribution. Given an external metric of importance, the autoassociative memory should be able to turn training on, off, or throttle it, ideally within the same framework. The model proposed in this chapter is a prime example of such a case. Biologically, it is known that memory storage and recall can be affected by factors such as saliency and emotion [156-158]; therefore, this type of biasing is a plausible line of thinking within the context of this research if the motion tracking layer is considered a memorization layer.

The application of importance biasing of pattern memorization in training an autoassociative network is a generalization of integrating retinal constancy to the motion tracking RBM. A novel contribution is made in the form of this importance biasing by using it as a coefficient for the learning rate such that training examples with higher coefficients impact the learning more than others; thus some patterns are weighted to be more important. Equation (4.18) shows the contrastive divergence weight update equation for the motion tracking layer, updated from Equation (2.16) to merge the concept of retinal constancy.

$$\Delta w_{ij} = \gamma \, r \, [p_{ij}{}^{+} - p_{ij}{}^{-}] \tag{4.18}$$

where γ is the learning rate, *r* is retinal constancy or importance coefficient, $p_{ij}{}^{+}$ is the positive association between visible node *i* and hidden node *j*, while $p_{ij}{}^{-}$ is the negative association. Execution occurs by presenting the motion vector and reconstructing the gaze movement vector, since desirable patterns are the only associations that are learned strongly, as follows: $h_{tracking}{}^{execution} = f_{tracking}(h_{motion}, 1)$.

Retinal constancy can be calculated using different representations of visual data. The first representation is to use raw visual data, discussed in section 4.1.5.1. Another is to use visual features instead of just pixels, discussed in section 4.1.5.2.

4.1.5.1 Raw Pixel Data

The most rudimentary representation used to calculate retinal constancy is simply the function of the difference between the raw pixel intensities of two images. When using image patches as the inputs for comparison, Equation (4.19) is used.

$$r\left(x^{(t_1)}, x^{(t_2)}\right) = max\left(0, 1 - \propto \sum_{i=0}^{N} \frac{\left(x_i^{(t_1)} - x_i^{(t_2)}\right)^2}{N}\right) \tag{4.19}$$

where *N* is the number of pixels in image patch *x*. The retinal constancy calculation essentially subtracts the coefficient-scaled mean squared error from 1, and stays 0 if the result is less than 0. The coefficient, ∝, can be considered the selectivity of the frame-to-frame comparison. Higher selectivity values give a lower tolerance, producing values of 0 more frequently, and only produce a high value when the frame-to-frame difference is small. As a result, the network will have fewer samples from which to associate eye movement to motion vectors, yet the accuracy of those samples will be higher. Conversely, lower selectivity produces fewer values of 0 and allows the network to see more samples, even if they are not necessarily good ones. Though there is no predefined range for the coefficient, the effective range depends on the number of pixels in the patch and the range of possible pixel values.

When using ideally simulated motion, there will not be any motion blur, noise, or variation from frame to frame. Therefore, a very high selectivity constant will suffice as frame-to-frame comparison will be nearly exact. In real-world data, if the selectivity constant is that high, samples that produce a sufficient retinal constancy value will never be found. As a result, it is important to adjust selectivity accordingly. The robustness of the selectivity coefficient along with the effectiveness of using raw pixel data for retinal constancy will be evaluated in section 4.2.7.

4.1.5.2 Visual Features

To mitigate some of the limitations of using raw pixel data, looking at more distinct features instead of just the raw pixels might be helpful and more robust. The features learned in Chapter 3 provide that type of robustness to recognize a visual pattern despite slight differences. It is possible that the invariant features are too robust that they may provide high retinal constancy despite an undesirable transformation. In this sense, invariant features are counterproductive to the goal. Here, a very high selectivity would be required to differentiate spatial transformations, since the features are robust to those transformations. This results in too few accurate training samples or, if the selectivity is too low, results in many inaccurate samples to the point that it would just be noise. This is demonstrated in section 4.2.7.

4.1.6 Motion Tracking Layer

The goal of the motion tracking layer is to associate the motion vector produced by the motion detection layer with the appropriate gaze movements, such that those gaze movements can be created by the motion vector. Effectively, this is a trained bi-directional autoassociative memory. First, this requires that the network learns the joint representation of the motion vector and gaze movements, formalized by Equation (4.3).

This association is learned by presenting the motion vector and movement vector as a single vector by concatenating the two, as shown in Figure 4.11. Biologically, this layer is similar to MST, since it has been shown to produce eye movements relating to pursuit.

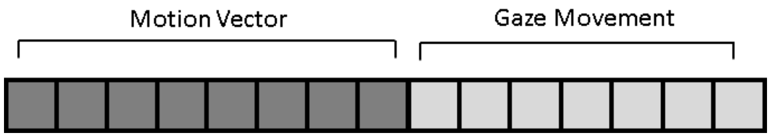


Figure 4.11. Example of the concatenated vectors that form the motion tracking layer visible vector.

The use of the concatenated vector and explanation of gaze perception and generation via the gaze movement vector as it relates to the motion tracking layer is shown in Figure 4.12.

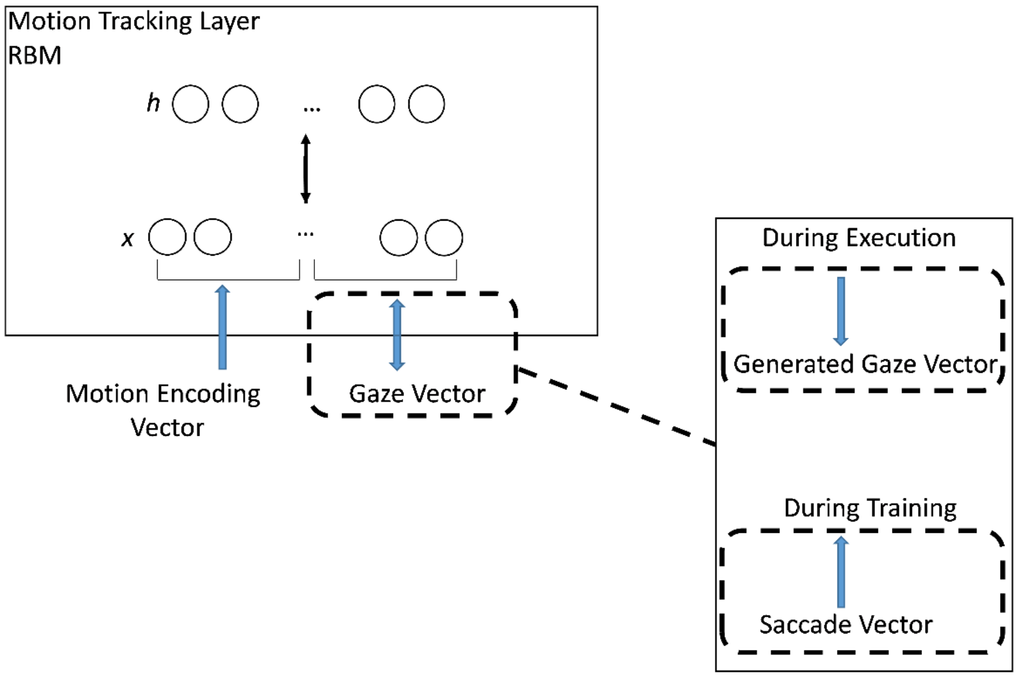


Figure 4.12. Motion tracking RBM.

From the perspective of the tracking layer, since there is no reasonable pattern that can be extracted from random gaze movements and independent saccade motion vectors, an additional piece of information must be given to the network to find meaningful patterns. The additional information can be considered an incentive, reward, or motivation, for the network. For motion tracking behavior to occur, the incentive is a value comparing the previous frame to the next frame. This incentive, known as retinal constancy, was described in section 4.1.5.

During training, the association of motion vectors to receptive field movements occurs. In a sense, it is a type of memorization or a method of storing patterns where certain motion vectors are stored along with corresponding eye movement vectors when they are considered important, based on the reward value. However, where this deviates from a simple memorization task is that the motion vectors are high-dimensional and many different potential vectors can correspond to the same eye movement. Also, simple transformations are not always encoded the same way and the network's response is at the mercy of the unsupervised learning algorithm.

During execution, the motion tracking layer behaves generatively to create the gaze. It is first primed by producing hidden activations based on the motion vector. Then, it uses those hidden activations to recreate the input gaze movement vector, which contains the appropriate gaze. This is formalized by the execution of one Gibbs step through Equation (4.5). Note that neither retinal constancy nor gaze is used as an input during execution, since only the motion vector is required to produce the appropriate gaze.

4.1.6.1 Timing

In the real world, physical limitations produce delays in various areas. To name just a few pertinent examples, delays exist in the propagation time between obtaining the visual information and processing, between processing and generation of control signals, and between the generation of the control signals and the actual movement of the gaze. As covered in section 4.1.3, in the proposed model, Δt_m is a functional delay, which is used to estimate motion. To practically simulate tracking, all of the other delays must be considered to form a cohesive theory.

There are two notable delays that must be defined here. The first delay is between motion encoding and the gaze action being generated by the motion tracking layer, considered as Δt_{g1}. The second delay is between the gaze action being generated and the gaze action reaching the target, considered as Δt_{g2}. This delay implies that some time must be taken for the gaze to shift to match the new position of the target. During training, the retinal constancy evaluation is performed between the visual contents of the receptive field at the start time, t, and at time $t + \Delta t_m + \Delta t_{g1} + \Delta t_{g2}$. During execution, the gaze will match the target position at $t + \Delta t_m + \Delta t_{g1} + \Delta t_{g2}$. Since the two individual delays do not need to be taken separately for the purposes of this work, Δt_g, henceforth named the *gaze delay*, is used to represent the sum of the two delays: $\Delta t_g = \Delta t_{g1} + \Delta t_{g2}$. The gaze delay is used to signify the propagation time between the output of the motion encoding module and the perception of the receptive field at the end of the gaze movement.

Assuming that the perception of the first frame is at time t, the second frame is at time $t+\Delta t_m$, the motion encoding perceives the target motion. Ideally, the saccade will catch up to the target, there will be a high retinal constancy, and the system will learn that the produced gaze movement is the right movement to make when presented with the given motion such that retinal constancy is maximized. The time it takes from the motion encoding to the evaluation of retinal constancy is represented as Δt_g. Thus, retinal constancy is evaluated at t+ Δt_m + Δt_g. Note that the definition of Δt_g only determines when the image is sampled for retinal constancy evaluation; as such it represents how quickly a saccade can move. The motion tracking layer itself does not know about this value, and does not base any calculations on the knowledge of this value. Therefore, it learns from whatever Δt_g is used in the model, which shows the robustness of the mechanism such that it can learn independently from whatever physical delays exist without requiring an update of the model. It does assume that the delay is the same during training and execution, which is to be expected. The timing relationships to the input, output, and the processing are shown in Figure 4.13.

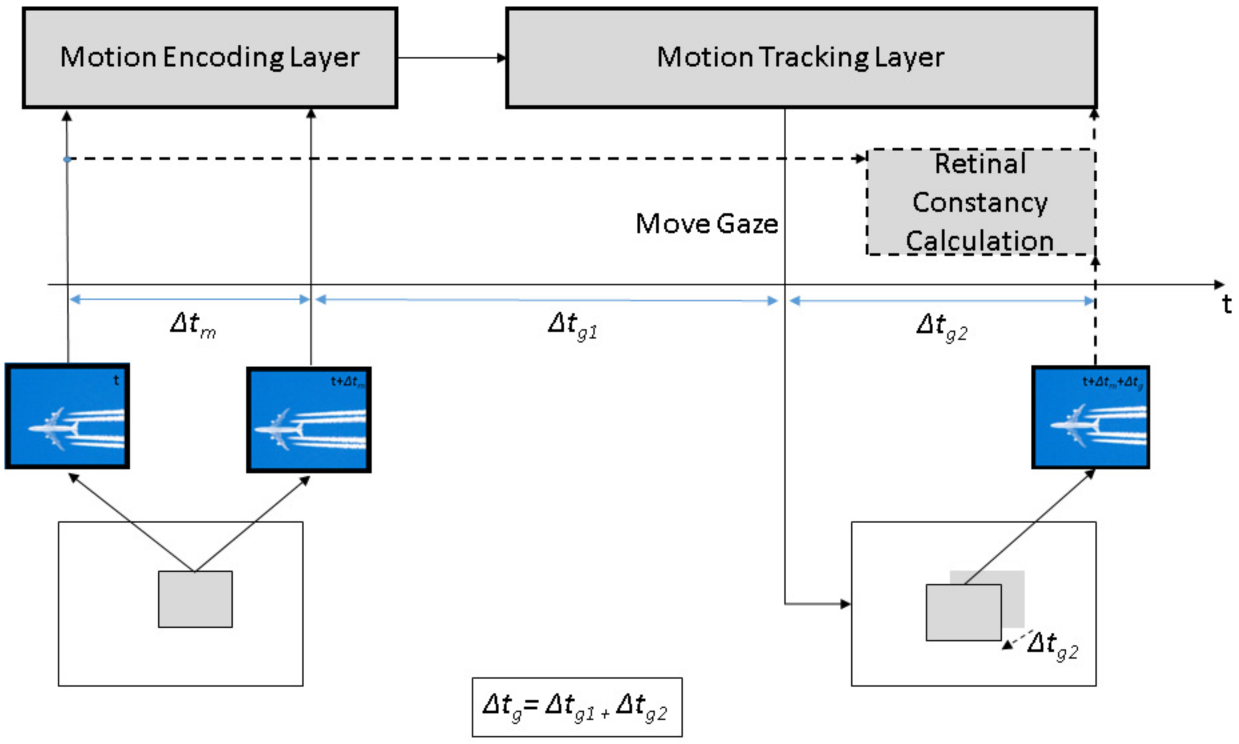


Figure 4.13. Timing and input/output diagram. Items using dotted lines occur only during training.

The plane moves to the left between time t and t+Δt_m. The network perceives this motion, and moves the receptive field to re-center on the plane at t+Δt_m +Δt_g; thus, the plane looks as though it is in a similar position in the receptive field as it was at time t. Effectively, the target's motion between t and t+Δt_m is used to estimate the target's position at t+Δt_m +Δt_g. Provided that the accuracy of the motion estimation is high enough, and that Δt_g is sufficiently small so that the velocity of the target does not change too drastically, the tracking will be effective.

4.1.7 Motion Detection, Visual Attention, and Compensation for Retinal Slip

An additional application of this motion tracking model is to behave as a motion detection model or a visual attention model.

4.1.7.1 Motion Detection

Since the model has a desire to move the gaze when a target is moving, it can be applied at positions covering the entire field of view, or video frame, to identify areas of motion between timesteps. Provided that there is a generated gaze other than (0,0), considered stationary, that position in the field of view can be considered an area of motion. Motion detection is determined by Equation (4.20).

$$\left(gaze_x{}^{(x,y,t)}, gaze_y{}^{(x,y,t)}\right) \neq (0,0) \tag{4.20}$$

where $\left(gaze_x{}^{(x,y,t)}, gaze_y{}^{(x,y,t)}\right)$ is the generated gaze at position *(x,y)* in the field of view, at time *t*.

4.1.7.2 Visual Attention

To go one step further than simply detecting motion, the outputs from the model can be used as labels to determine how strong the perceived motion is. If the areas are labeled with the activation values corresponding to that gaze movement, these can be considered as saliency values, as shown in Equation (4.21). That value portrays how strongly the network responds to the features and motion in that area. The concept of saliency here refers to how interesting an area is to the model, based on the visual motion features and gaze movement correlations that it has learned, and it differs from other computations of saliency which rely on contrast, colour, orientation, and motion, such as [139, 140].

$$\beta(x,y,t) = \begin{cases} \left(gaze_x{}^{(x,y,t)}, gaze_y{}^{(x,y,t)}\right) \neq (0,0) & \beta' \\ otherwise & 0 \end{cases} \tag{4.21}$$

$$\beta'_{axis-split} = \max\left(gaze_{horizontal}{}^{(x,y,t)}\right)^2 + \max(gaze_{vertical}{}^{(x,y,t)})^2$$

$$\beta'_{full\,position} = \max\left(gaze_{2D}{}^{(x,y,t)}\right)$$

If there is detected motion at position (x, y) at time t, as in Equation (4.20), the saliency β is calculated for that position. If using the axis-split gaze movement vector representation, the sum of the squared activation values corresponding to the chosen horizontal and vertical gaze is the saliency, $\beta'_{axis-split}$. If the full position gaze movement vector representation is used, the saliency becomes the activation value of that position, $\beta'_{full\,position}$.

Applied to each position in the field of view, a saliency map can be created. A ranking of these saliency values will give the location with the highest saliency, effectively behaving as visual attention by providing an order by which to visit these areas.

It should be noted that the model's response does not depend solely on motion. It is strongly influenced by motion, but there are also some patterns it responds to more than others; as a result, this visual saliency combines visual feature information with motion information, making it a more complex calculation than a basic motion detector.

The application of this in real-world video will be demonstrated in section 4.2.9.

4.1.7.3 Compensation for Retinal Slip

As discussed in section 2.2.6.3, retinal slip occurs in smooth pursuit when the target is not tracked accurately. Sometimes, the target may move slightly unpredictably, or the motion perception will perceive it slightly inaccurately, such that the gaze falls behind or ahead of it during tracking. Also, after a period of tracking, based only on motion perception, it is inevitable that there will be a drift over time [159].

The current model is unable to compensate for slip on its own, since it does not perceive anything outside of its receptive field, and only knows about how the pattern inside it has changed between timesteps. If the pattern changes because the target has moved outside of the receptive field, the model only knows how to follow that new pattern, which may simply be the static background.

However, with a little bit of information from outside of the receptive field, and applying the visual attention model described in section 4.1.7.2, retinal slip can be compensated for. After slipping occurs, the visual attention model can be engaged to pick the most salient target within the region surrounding the receptive field. Ideally, this will select the same target again, provided it has not moved too far out of the area surrounding the receptive field or that another more salient target does not enter that same area.

In this thesis, retinal slip compensation will be engaged when there is no motion for a period of time. This will be demonstrated in section 4.2.10. More complex methods of engagement can be used, such as determining when the contents of the receptive field differ too much from the tracked patch, similar to adaptive background subtraction [160]. Such a method would also be robust to a non-stationary background or a new target preventing the engagement of the compensation. However, those concepts are beyond the scope of this thesis.

4.2 Experimental Results

The network described in section 4.1 is implemented, trained, and tested. The details of those three stages are outlined in this section. Initially, the code was written in C++ with the use of the OpenBLAS library [161] to speed up matrix arithmetic calculations. OpenCV [162] was used for display purposes. Later, an implementation was made in Python using NumPy and SciPy [163] to efficiently compare testing scenarios.

4.2.1 Implementation Details, Training, and Testing Overview

The learning process is carried about by greedily training the system layer-by-layer. Each layer is trained from the bottom up, starting with the motion encoding layer and ending with the motion tracking layer. Upon training the GRBM, its hidden activations are used as inputs to the tracking layer. The tracking layer also takes in the gaze movements as inputs and throttles the learning by the retinal constancy value, as described in section 4.1.5.

4.2.1.1 Motion Encoding GRBM

Receptive field sizes were fixed to 14x14 pixels; the size is large enough that there is enough information to disambiguate motion at an accuracy high enough for this thesis, but small enough that it is not computationally taxing. The model operates on videos of different sizes and tracks objects that are of different scales; this means that the object does not necessarily have to fit within the receptive field, and that only a strong feature must be present within the receptive to facilitate tracking.

During training, a gaze movement is generated with probability *p=0.5*, and the gaze movement is randomly selected from the set of possible transformations shown in (4.10), where each possible shift has an equal probability of being chosen. Using a probability of *p=1* would speed up the training, however, a lesser probability is chosen to show the robustness of the system to having fewer useful training samples. Also, though it is trivial, learning that having no gaze movement corresponds to a motion vector that encodes no transformation is important for the model to learn when not to change gaze, or in technical terms, produce a gaze movement of (0, 0).

The GRBM motion encoder has 196 visible nodes, to support the 14x14 receptive field defined above, for the frame at *t*, and naturally, 196 visible nodes for the frame at $t+ \Delta t_m$. 200 factors are used, along with 100 hidden nodes. This is the configuration used for the tests unless otherwise stated. The outputs of those 100 hidden nodes will be the motion vector fed to the motion tracking layer above. 100 hidden nodes and 200 factors were determined experimentally to produce the highest discriminative results. Contrastive divergence, with a learning rate of 0.01, a sparsity of 0.2, and batch sizes of 100, was used for training the motion encoding layer.

All visual data is converted to grayscale before being passed to the motion encoding layer. For preprocessing, a neighbourhood size of 1 is used for the local contrast detection, based on Equation (4.7). Contrast normalization is performed based on Equation (4.8). Also, the same input blurring is used as in 3.1.4. The combination of local contrast detection, contrast normalization, and input blurring was found to increase discriminative performance of the GRBM by over 80% when compared to using raw pixel data in experimental work in support of this thesis, which is not covered here.

Since the two-layer network is too small to be discriminative, sufficiently encode most real-world transformations, and encode correlations to all possible gaze movements within a large visual space, the simulations enforce some practical limits on Equation (4.10). It limits both the magnitude of the gaze control and, correspondingly, the magnitude of possible transformations to encode.

$$\Gamma_1 = \{(r,s) | (r,s) \in \ ([-1,+1],[-1,+1])\} \tag{4.22}$$

$$\Gamma_2 = \{(r,s) | (r,s) \in \ ([-2,+2],[-2,+2])\} \tag{4.23}$$

In all experiments, this corresponds to having gaze control being limited to either -1 to +1 pixels or -2 to +2 pixels both horizontally and vertically, where $\text{gaze} \in \Gamma$. Since these are the limitations of the gaze control, it makes sense to also limit the possible motion the network is trained with such that it is not presented with motion that it cannot track due to its limited gaze range. Where possible, meaning in synthetic motion, the network is only trained on motion that occurs within -2 to +2 pixels both horizontally and vertically between frames t and $t+\Delta t_m$. When using video, no such motion limitation is applied during training or execution; however, the network can still only encode a limited gaze.

Finally, since there is no filtering used for tracking, the network would rely solely on motion encoding from two sequential frames. It is well-established in literature that noise and inaccuracies in object detection from frame to frame can cause significant error in motion tracking without filtering. To increase the stability without carrying over information to subsequent timesteps, a convolution procedure is proposed. Instead of relying only on the window on interest, a larger window of information is used based on the idea that similar motion is likely to occur in neighbouring regions. The motion encoding is executed on the visual data in the receptive field as well as the visual data surrounding it, and a pooling operation is performed on the resulting encoding vectors. In this implementation, the nearest non-overlapping neighbours are used as the surrounding visual data, resulting in eight surrounding windows that are the same size as the receptive field. The receptive field, and each of the eight surrounding windows, are fed to eight identical copies of the motion encoding layer in parallel, behaving as a convolution kernel. The average pooling operation, from Equation (2.34), is applied on the resulting motion encoding vectors to produce the final motion encoding vector which is propagated upwards. This process is shown in Figure 4.14.

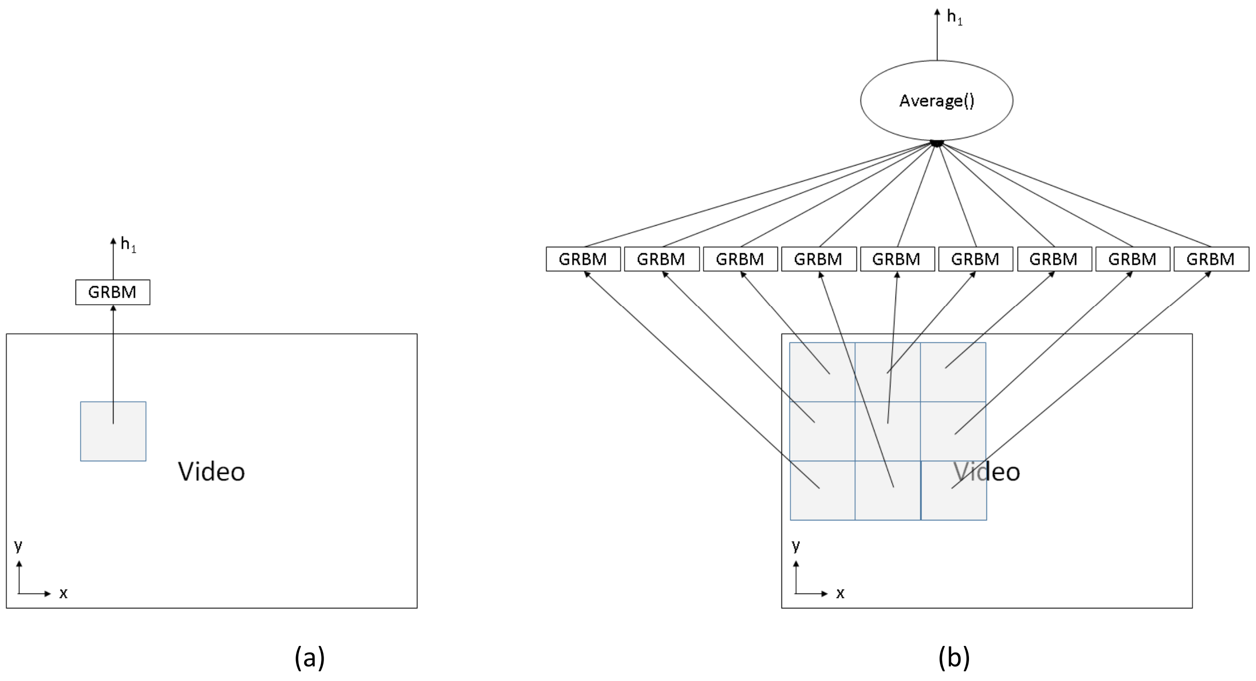


Figure 4.14. (a) Motion encoding without spatial pooling, and (b) motion encoding with spatial pooling. Eight identical copies of the GRBM are used, one for each window, in parallel, behaving as a convolution kernel.

This increases the motion encoding accuracy significantly since it uses more surrounding information to reduce the impact of resolution, noise, and calculation inaccuracy that would greatly affect just a single sample. In a sense, this can be considered spatial filtering instead of temporal filtering.

Whenever it is not specified, the motion encoding GRBM is trained using synthetic shifts of grayscale patches from the van Hateren natural image database, considered synthetic motion. The performance of synthetic motion is compared to the performance of real-world motion in section 4.2.4.

4.2.1.2 Motion Tracking RBM

Contrastive divergence is used to train the motion tracking RBM, with a learning rate of 0.01 and no sparsity. The number of visible nodes in the RBM is the sum of the number of hidden nodes in the motion encoding layer and the gaze movement vector size, which depends on the gaze range and the encoding type, as described in section 4.1.4.

The motion tracking hidden layer contains only 25 nodes. The number of hidden nodes is kept small such that the network must be efficient about the correlations it learns. With a larger number of hidden nodes, it is likely that trivial correlations will be learned. For example, individual hidden nodes will come to represent individual visible nodes in the gaze movement vector. Therefore, the initialization of the gaze movement vector prior to execution will too strongly influence the reconstruction of the gaze movement vector, instead of the desired effect of having the motion vector influence the reconstruction of the gaze movement vector. For example, with a large number of nodes, despite the motion vector, when the gaze movements are initialized to some small value near 0, when the RBM is executed and the gaze movements are generated, they will remain as 0s instead of changing based on the associations it should have learned. Same with an initialization of 1s. This makes it impossible to find a good initialization value for the gaze movement vectors prior to reconstruction.

A small number of hidden nodes seemed to suffice with a motion vector of 100 nodes. This indicates that produced motion vectors are very similar for motion of a certain direction and speed.

All visual data is converted to grayscale before being used for retinal constancy calculations. Also, random saccades, as described in section 4.1.2.1, are used for training unless otherwise noted.

4.2.1.3 Tracking Accuracy Measurement

The tracking accuracy was calculated by how accurately the model tracks artificial motion. A number of translated image pairs are fed to the model and it attempts to produce the correct gaze control to nullify the translation. Random high-variance patches are selected from the van Hateren natural image dataset. A translation is artificially generated for each of those, randomly chosen from the limited subset of allowable transformations in Equations (4.22) or (4.23), thus the ground truth is known. The result of the model's predicted translation and the ground truth is compared over all patches to determine the tracking accuracy. A tracking accuracy of 1.0 means that all transformations were correctly predicted. Note that training occurred with randomly selected patches in each of the 4000+ high-resolution images, so it is unlikely that there is an overlap between the testing data and the training data. 50,000 image pairs were used for training and 1000 image pairs are used for testing. High variance patches are used so that there are sufficient high contrast features for the motion encoding layer to behave effectively. The search is conducted by setting a threshold, 5.0, and the first random patch that has a variance exceeding that threshold is chosen as a high-variance patch.

4.2.2 Vector Representation Comparison

As mentioned in section 4.1.4, the motion vector can be represented as a full binary vector, an axis-split binary vector, or an analog vector. The full binary vector encodes every possible position as a different bit in the binary vector, making it a better discriminative solution. However, it is the least compact solution, and suffers from an exponential increase in representation size as the range increases. The more realistic analog vector attempts to simulate muscle control, does not increase with range size, and can leverage limited training data due to the same nodes learning from nearby patterns in feature space. However, the behaviour of an analog node is more unpredictable, especially in a network designed for binary states. Finally, the axis-split binary vector is a balance between the two solutions, where there are different bits for each position along an axis. This representation increases linearly with the range size, and aims to balance discriminative power with leveraging limited training data. Figure 4.15 shows the performance of the network using the different representations in a [-1, +1] pixel movement range.

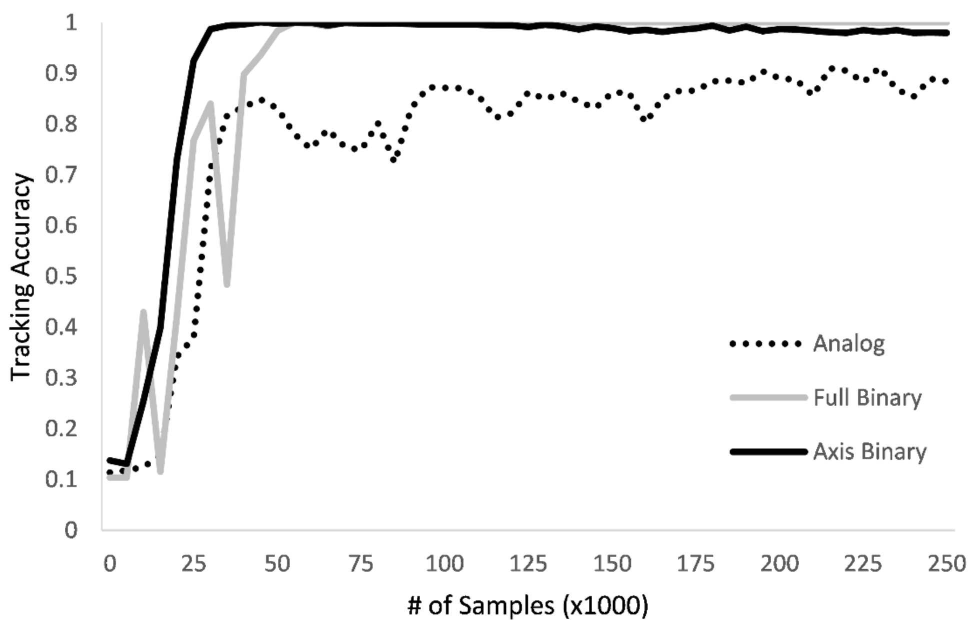


Figure 4.15. Tracking accuracy vs # of training samples over a [-1,+1] pixel movement range for a network with different gaze movement vector representations: analog, full binary and axis-split binary.

The full position and axis-split binary representations reach a near perfect accuracy, but the analog representation lags behind. However, the analog vector still performs impressively considering that it encodes 3 values (-1, 0, +1) in each node instead of just one value like the other representations. Also, the axis-split binary representation reaches peak performance in half the number of samples as the full position binary representation, meaning that it can better leverage limited training data. Figure 4.16 shows the performance of the same representations in the [-2, +2] pixel movement range.

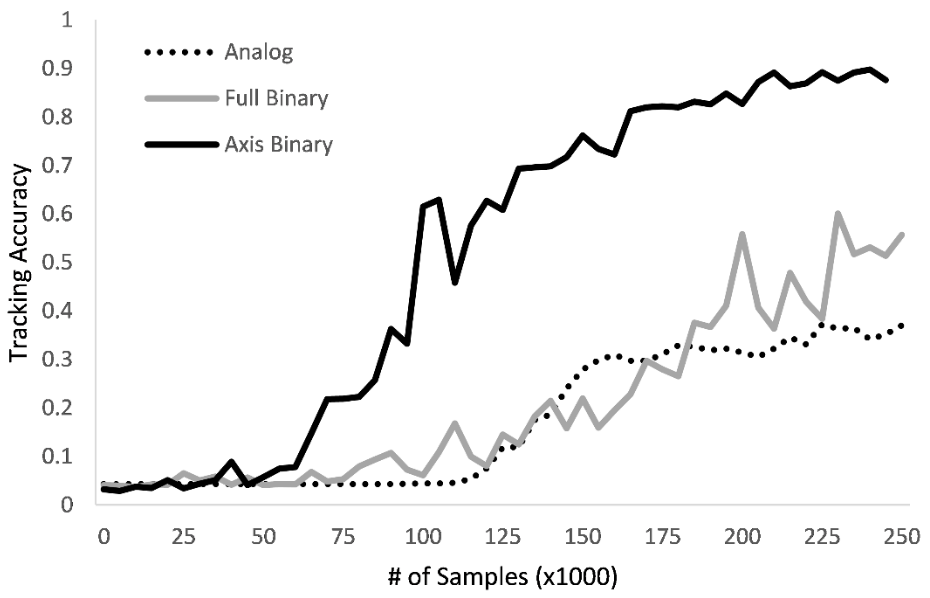


Figure 4.16. Tracking accuracy vs # of training samples over a [-2,+2] pixel movement range for a network with different gaze movement vector representations: analog, full binary and axis-split binary.

The application of these representations on the [-2, +2] pixel range gives more interesting results about their scalability and stability. The performance of the analog and full position representations decreases quite a bit when trying to encode larger ranges, while the axis-split binary representation does not lose much of its accuracy. The analog representation is trying to encode 5 values (-2, -1, 0, +1, +2) into a single node, which explains its decrease in performance. However, at a peak performance around 40%, that is still much better than randomly guessing the correct transformation at 4% according to Equation (4.13); this is likely due to its ability to leverage the limited training data. The full position representation does get higher results, but is a less impressive result since it can encode each possible position without any overlaps. Its inability to leverage data from similar, but inexact, patterns makes it much harder to get enough training data to cover each of the possible patterns. Based on these observations, though the network is robust enough to withstand different representations, the axis-split binary representation is the best compromise between compactness and discriminative ability; it also performs significantly better than the others. Performance aside, the proof of concept in the analog representation shows that it is indeed possible to represent analog muscle control in such a model. With a more advanced neural model, especially one designed specifically for analog data, an analog representation would be much more effective. Since the split-axis binary representation is the best performing representation, all further experiments will be using that representation.

4.2.3 Qualitative Analysis

Figure 4.17 shows examples of transformations that the network correctly tracks. These positive examples show how the network is able to correctly track some patches which have patterns that are very difficult to discern. Figure 4.18 shows some examples that the network could not track correctly. The examples are arranged by showing the raw patches that are fed to the model, the effects of preprocessing described in section 4.1.1, and the actual motion compared to the model's predicted motion.

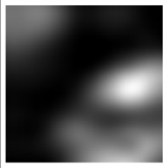
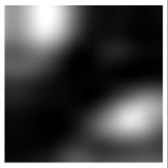
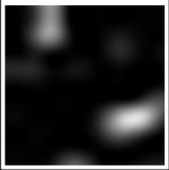
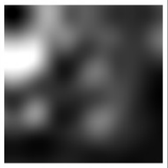
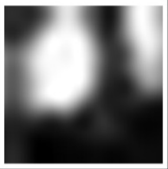
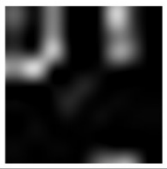
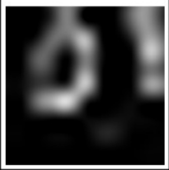
Raw Patches		**Preprocessed**		**Motion**	
t	t + Δt_m	t	t + Δt_m	Actual	Predicted
				(0, 2)	(0, 2)
				(0, 0)	(0, 0)
				(2, 2)	(2, 2)

Figure 4.17. Selected examples of sequential patches taken from a video that the network tracks correctly. Each row is a separate example.

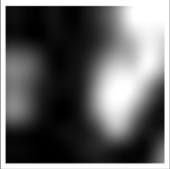
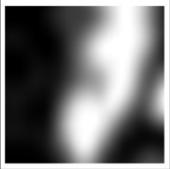
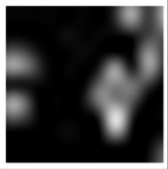
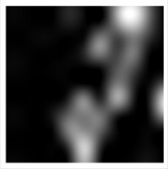
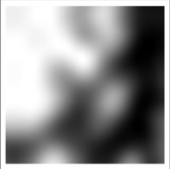
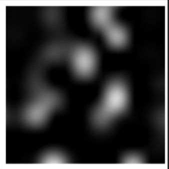
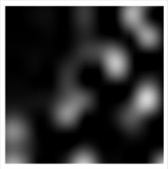
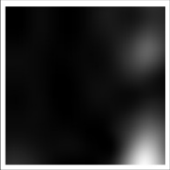
Raw Patches		Preprocessed		Motion	
t	$t + \Delta t_m$	t	$t + \Delta t_m$	Actual	Predicted
				(-2, 2)	(-2, 1)
				(2, 0)	(2, 1)
				(-1, 2)	(0, 2)

Figure 4.18. Selected examples of sequential patches taken from a video that the network does not track correctly. Each row is a separate example.

These examples, both successful and unsuccessful, are difficult to draw conclusions from. The network's motion features adequately describe the translation of one pattern but not necessarily the other, even if they undergo the same transformation. The black box nature of neural networks makes it difficult to diagnose exactly which patterns will be effective. It is likely that the most important pixels for a given pattern are located at coordinates that have not developed strong connections to the corresponding motion feature due to the training data. A more valuable example would be how the model works in regular video as opposed to selected patches, and this will be covered in section 4.2.8.

4.2.4 Synthetic Motion vs Real-World Video

Figure 4.19 shows the model's tracking accuracy when trained on synthetic motion and when trained on real-world video. It can be seen that the synthetic motion is far more effective at training than the real-world video. The same figure shows the model's performance when capable of a motion range of -2 to +2 pixels, compared to a motion range of -1 to +1 pixel. A smaller motion range means that the network is likely to make fewer errors, since there are fewer possibilities in the gaze movement vector; 9 for a -1 to +1 pixel range, and 25 for a -2 to +2 pixel range. For example, a random selector would succeed with an accuracy of 11.11% for the -1 to +1 pixel range, while it would only succeed with an accuracy of 4% for the -2 to +2 pixel range based on Equation (4.13). The gaze range increases the number of possibilities in the gaze movement vector exponentially. As a result, there are fewer training samples for each motion. Thus, it is no surprise that there is a notable accuracy difference between methods using two different motion ranges.

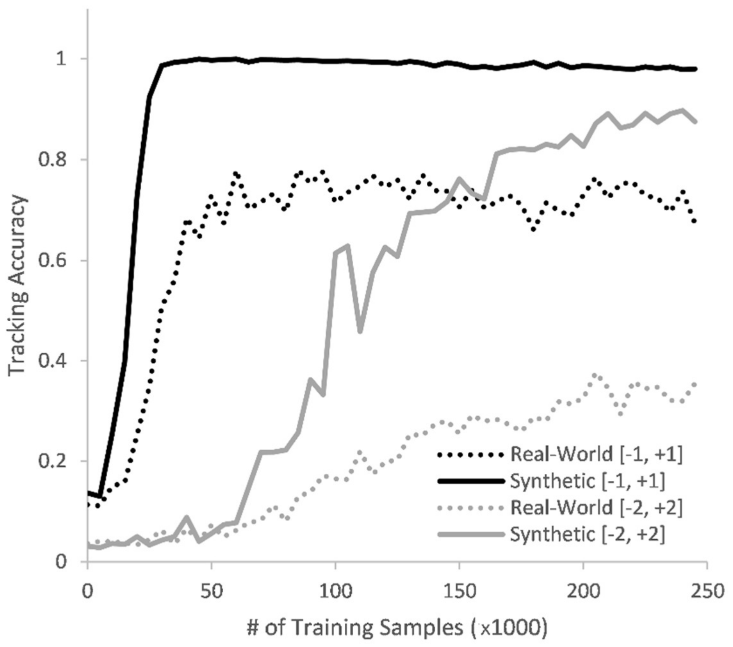


Figure 4.19. Tracking accuracy vs number of training samples of the network, comparing real-world and synthetic motion within [-1,+1] and [-2,+2] pixel motion ranges.

4.2.5 Motion Encoding

First, the motion encoding GRBM was trained with the grayscale version of sequential patches in the Hollywood2 video data. In terms of the sampling frequency required to detect motion, Δt_m should be sufficiently large that motion can be detected, yet sufficiently small that it can be analyzed quickly and propagated for further processing. In this implementation, Δt_m = 1 frame, so two consecutive image frames are used to estimate motion: a frame at time *t*, and a frame at time *t+1*.

Interpreting visualizations of GRBM filters are slightly different than interpreting the image feature representations shown in Chapter 3 since the higher-order nature of the network makes it difficult to easily visualize. An ideal visualization requires the difficult task of visualizing which pixel in the input image corresponds to which pixel in the output image. A simpler visualization is chosen here. For the purposes of image transformations, the GRBM visualizations will represent the factors, not the hidden nodes. Each factor can be described by a pair of images, where one image describes what connections that factor has to the input nodes and the other image describes what connections that factor has to the output nodes. This will give an indication of what areas of the input image and what areas of the output image inhibit or excite the factor. Similar to the image feature visualizations, brighter areas are excitatory, darker areas are inhibitory, and grey areas have no impact. The relationship between the factor's connections to the input nodes and the factor's connections to the output nodes explain what type of transformation it responds best to. For example, a striped pattern from the input nodes corresponding to a shifted striped pattern from the output nodes can be interpreted as that factor encoding that shift. In an effort to visualize all of the factors, two grids of images will be shown side by side. An image in one cell of the left grid represents a factor's connection to the input nodes. An image in that same cell of the right grid represents that same factor's connection to the output nodes. After training the motion encoding layer on 5 million samples, the factors learn to resemble 2D shifts in a variety of directions. A visualization showing the learned transformations of each of the hidden nodes is shown in Figure 4.20. GRBMs have factors responding to a pair of inputs, thus, the visualizations are composed of two sub-images: one for the first input nodes, *x*, in Figure 4.20a; and one for the output nodes, *y,* in Figure 4.20b. Each square in a sub-image represents a different factor (200 in total for the GRBM), and the squares in the same position on both sub-images represent the same factor. The pattern shown effectively represents how the factor is connected to the input nodes, thus, corresponding squares in both images represent how that given pattern should be transformed in the output nodes to elicit a response from the corresponding factor. The 2D arrangement of the factors is only due to space constraints, the network does not have any 2-dimensional topographical structure.

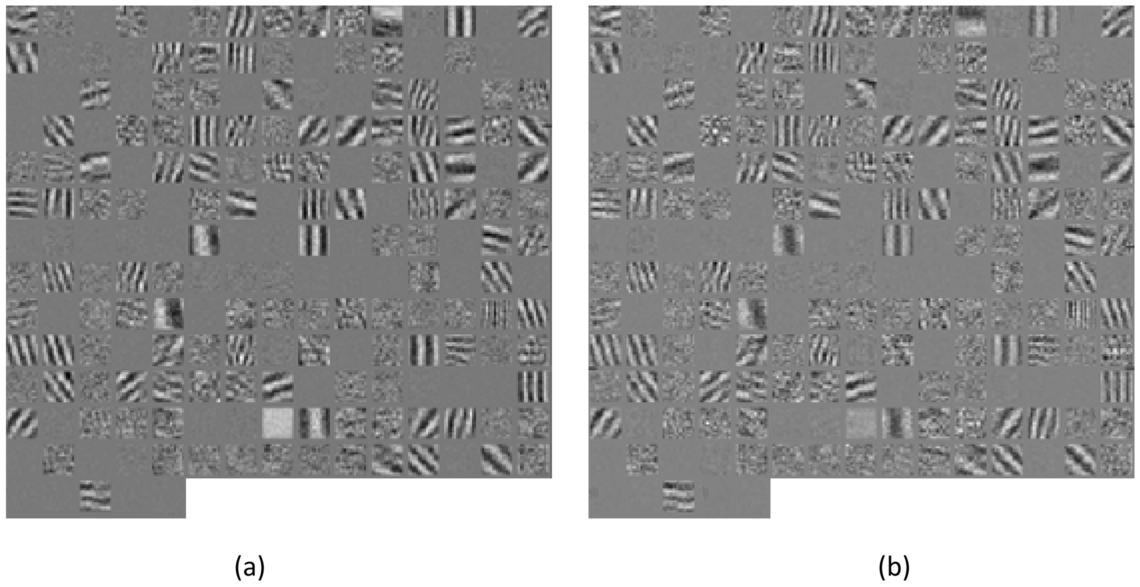
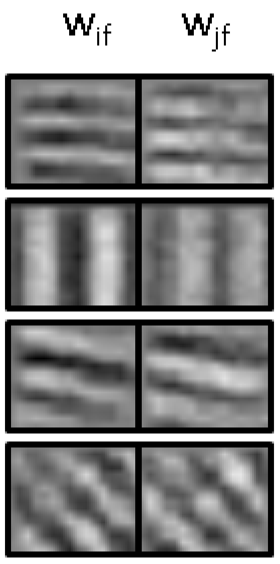


Figure 4.20. Learned GRBM factors; (a) frame 1, or x(t), weights corresponding to w_{if} ; and (b) transformed frame 2, or x(t+Δt_m), weights corresponding to w_{jf} .

From the examples, it can be seen that not all of the factors come to represent something valuable. Many learn noise or are empty, which means the motion is described by only a subset of the 200 factors, and only those are usable to describe the motion. However, it is also notable that many of the factors come to represent different translations, a few of which are shown in Figure 4.21. It is possible that the subset represents all possible transformations sufficiently. It is also possible that the network could not learn filters for reasons such as a lack of training iterations or data complexity. To determine if the network sufficiently represents the transformations, its discriminative performance must be tested.



(a) (b)

Figure 4.21. Examples of some learned transformations. Each row represents a different learned factor, with (a) the frame 1 pattern, or x(t), corresponding to w_{if} ; and (b) transformed frame 2 pattern, or x(t+Δt_m), corresponding to w_{jf} .

It is important to evaluate the effect of the motion encoding layer size on the motion tracking model's accuracy. It is thought that a larger motion encoding network would allow a higher tracking accuracy due to its ability to represent more patterns; this is examined by modifying the model's default GRBM parameters, retraining, and measuring performance relative to these parameter changes. Figure 4.22 shows that a larger motion encoding hidden layer is more effective at tracking to some extent, but then levels off when a certain critical size is reached. Increasing the number of factors, while keeping the number of hidden nodes constant, improves performance, yet after 50 factors the performance gain is not as pronounced. However, an increase in hidden nodes, above a certain required minimum, does not improve performance as shown in Figure 4.23. In a nutshell, more factors produce better results, but more hidden nodes do not. Also, as shown in section 4.2.4, the van Hateren data with synthetic motion produces better results than the Hollywood2 dataset with real-world motion, even under varying factors and hidden nodes in the GRBM. The synthetic motion does not contain the motion blur or more complex translations found in real-world video, therefore, they provide much cleaner training data for the network to learn translations from. With higher resolution, higher frame rate video, it is likely that the gap in motion feature learning between synthetic and real-world video can be closed.

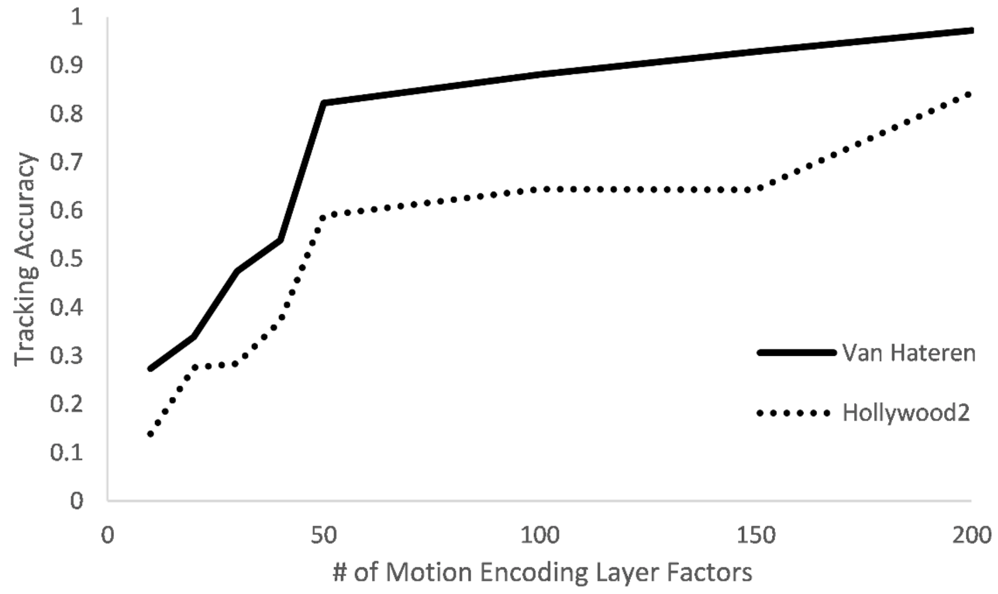


Figure 4.22. Tracking accuracy vs number of factors in the motion encoding layer.

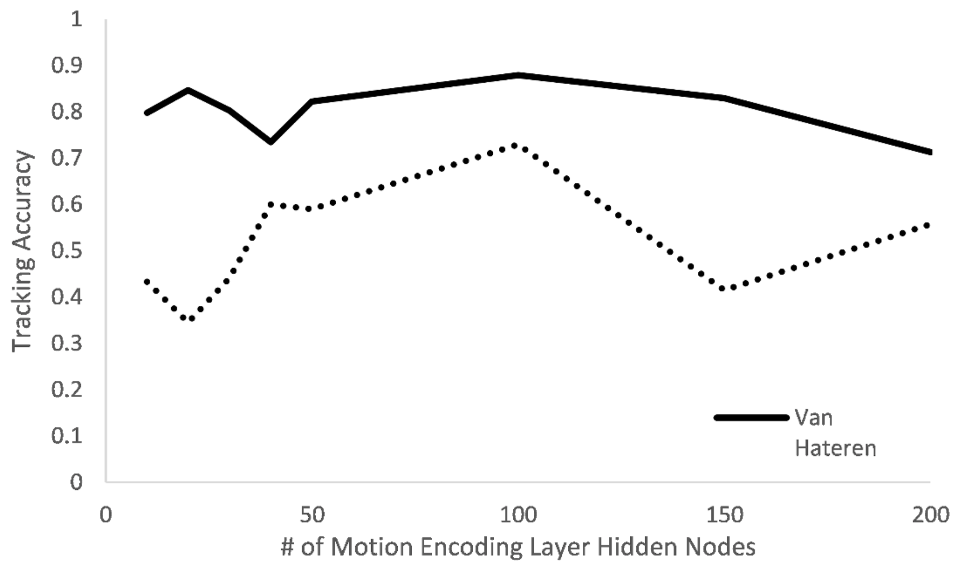


Figure 4.23. Tracking accuracy vs number of hidden nodes in the motion encoding layer.

The feature detectors developed by other methods from section 3.1 were also tested to determine whether they are effective. However, none of those methods provide any increase in performance. Given the results, the GRBM used for the motion encoding layer has 200 factors and 100 hidden nodes, as that is where peak performance was found over the network parameters tested.

4.2.6 Saccade Generation

As discussed in section 4.1.2, saccades can be generated either as an external random process, separating saccade and motion tracking systems, or one that is driven by the motion tracking system itself. The evaluation here is to determine if a random walk through the search space would provide better results than a guided sample selection.

While using the motion tracking system to drive the gaze generation during training also, it is a more complete self-organizing system, yet theorizes that the motion tracking can organize without external influence. However, using the external saccadic function lends more credence to the biological theory proposed here that one independent system influences the other.

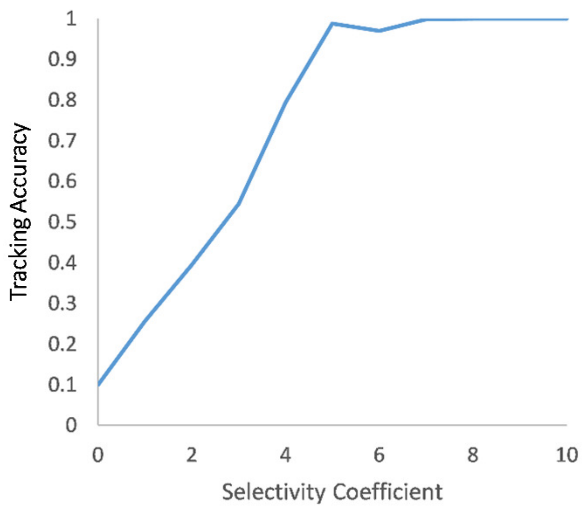
Method	**[-1, +1] pixel range**	**[-2, +2] pixel range**
Random Saccade	0.98	0.90
Motion-driven Gaze	0.99	0.90

Table 4.1. Tracking accuracy of random saccade gaze generation vs motion-driven gaze generation at [-1,+1] and [-2,+2] pixel ranges after 250000 training samples.

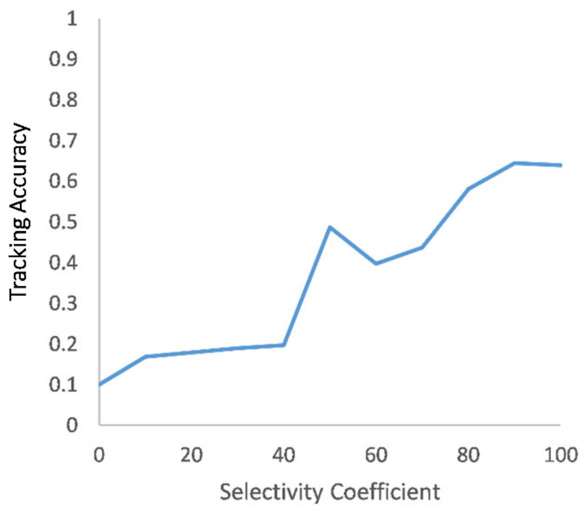
Table 4.1 shows that both techniques produce a similar result in terms of the tracking accuracy, defined in section 4.2.1.3. From a computational perspective, the random saccade generation is simpler in that it does not need any calculated input to generate the saccade. However, from a theoretical perspective, both show that a similar self-organization can occur despite differing models. Regardless, there is no practical benefit to one over the other; as a consequence, all results outside of Table 4.1 assume the use of random saccade gaze generation during training.

4.2.7 Visual Feature Representation vs Retinal Constancy Selectivity

As discussed in section 4.1.5, there are many ways of representing visual data for the purpose of retinal constancy. Here, we compare only raw pixel data and visual features learned by an RBM. The visual features are learned by an RBM using the same principles as in section 3.1.1; it is a sparse RBM with no additional regularization that has been trained for 200 epochs on random patches from a grayscale version of the CIFAR-10 dataset. The tracking accuracy is evaluated based on the network learning on increasing selectivity coefficients, $\propto$ from Equation (4.19), over 250,000 training samples and comparing the two visual representations. The results are shown in Figure 4.24.



(a)



(b)

Figure 4.24. The effect of the selectivity coefficient on the model's motion tracking accuracy when using retinal constancy: (a) based on raw pixels, and (b) based on learned visual features. Note that the x-axis in the raw pixel graph goes from 0 to 10 and the x-axis in the visual feature graph goes from 0 to 100.

It can be seen from the results that raw pixel data is far more effective. The steady increase in performance with selectivity coefficient, up to $\propto=5$ where it plateaus at maximum performance, shows that the network is most robust to selectivity coefficients when using raw pixel data. The visual features do not fare so well; though the network performance does steadily increase, it takes the network a much larger selectivity constant and still only reaches a performance of 0.6. This is because the visual features serve the purpose of describing visual data with robustness to translation and rotation, and the inherent simplification of the feature space requires the loss of the pixel-level precision required for proper motion discrimination. Raw pixels are quite obviously better, but the accuracy shown in the graphs may not correspond to performance in real-world video, as these classification values are calculated based on synthetically shifted real-world images to simulate motion. The evaluation of the network on real-world video will be performed in the next section. All results outside of this subsection use raw pixels as the feature representation.

4.2.8 Real-World Tests

For a more realistic execution, which is one of the main goals of this thesis, the proposed model was also tested on several pieces of real-world video footage. For these tests, a target was manually selected at the beginning of the sequence, using a Δt_m of 1 frame and a [-2, +2] pixel range axis-split binary gaze representation. In this implementation, Δt_g is set to 1 frame, meaning that the gaze is driven to the predicted location of the target in the frame after when the motion is perceived. Larger values of Δt_g mean that there is more room for error, since the target has more time to change its motion unpredictably before the gaze is driven to it. The model was only trained once. The motion encoding layer was trained according to the process outlined in section 4.2.5, using synthetic motion on the van Hateren natural image dataset. The motion tracking layer was also trained using synthetic motion on the van Hateren natural image dataset over 200,000 iterations. By that point the network was able to learn strong associations between the gaze movement, the high retinal constancy value, and the motion vectors. The same network was then used for all test cases.

Figure 4.25 shows selected frames from a video of the network tracking a truck on the highway. The truck, starting in the top left corner of the image, is tracked over the course of 115 frames, ending near the top right corner of the image. There is some slipping and some jitter, however it is minimal, and it is not so detrimental that the truck falls entirely out of the receptive field. Also, it tolerates the gradual scaling of the truck as it approaches the camera.

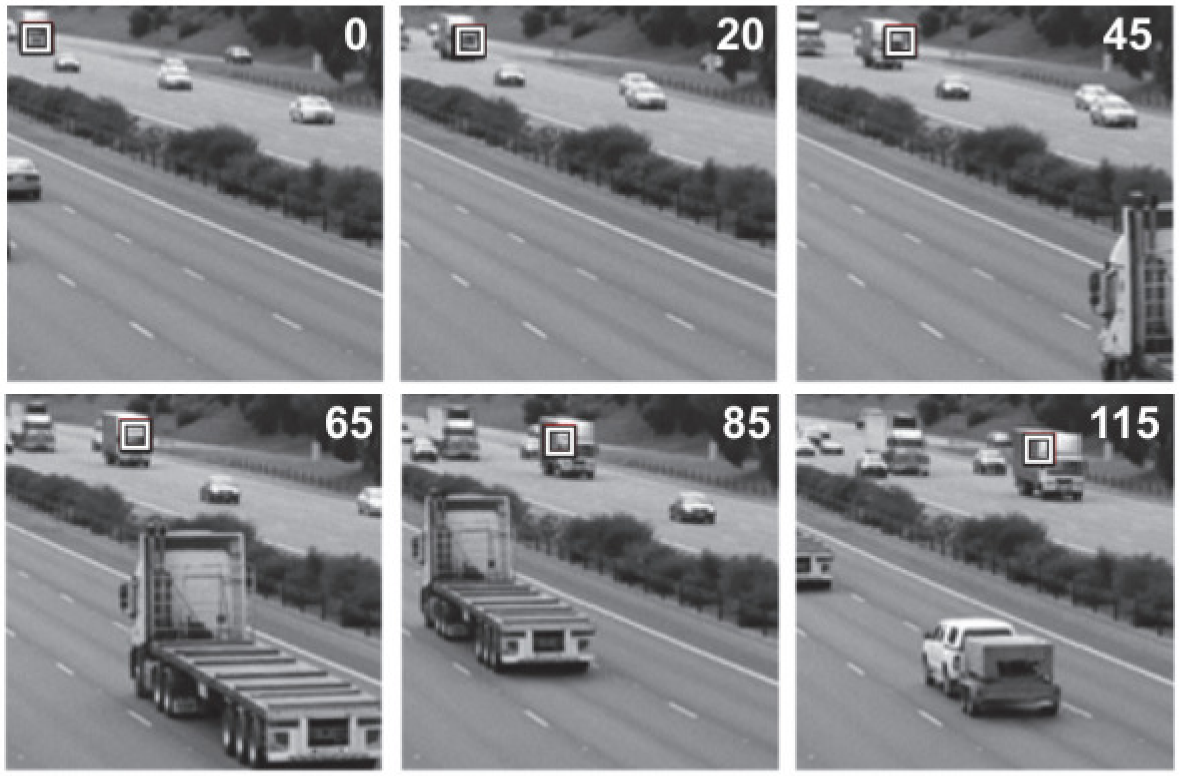


Figure 4.25. Selected frames of the network tracking a truck in highway video footage with the frame number listed in the top-right corner. The gaze generated by the tracking RBM is represented by the black and white square.

The second example, shown in Figure 4.26, exhibits a more complex motion with an object that significantly changes scale by showing a perspective of a soccer ball being shot away from the camera. Though the network is not designed to handle scale changes, it is robust enough to track different types of patterns, whichever ones happen to appear in the receptive field. This shows the network's ability to follow the target on a more complex motion, including a changing pattern due to ball rotation, a changing scale due to distance, and illumination changes. It starts by following a piece of the ball, and then the entire ball as the view of the ball decreases in size so it can fit inside the receptive field. This tracking occurs over the course of 150 frames.

The third video is that of a flock of birds, which shows the network's ability to track an object that changes shape and its ability to tolerate overlapping motion, performed over 200 frames. The network follows a single bird through a curved trajectory and is shown in Figure 4.27. In the 90th frame, there is an overlap with the motion of other birds. Also, the bird's shape changes due to flapping wings and direction change. The network, despite slipping and following different features of the bird, still manages to track its motion.

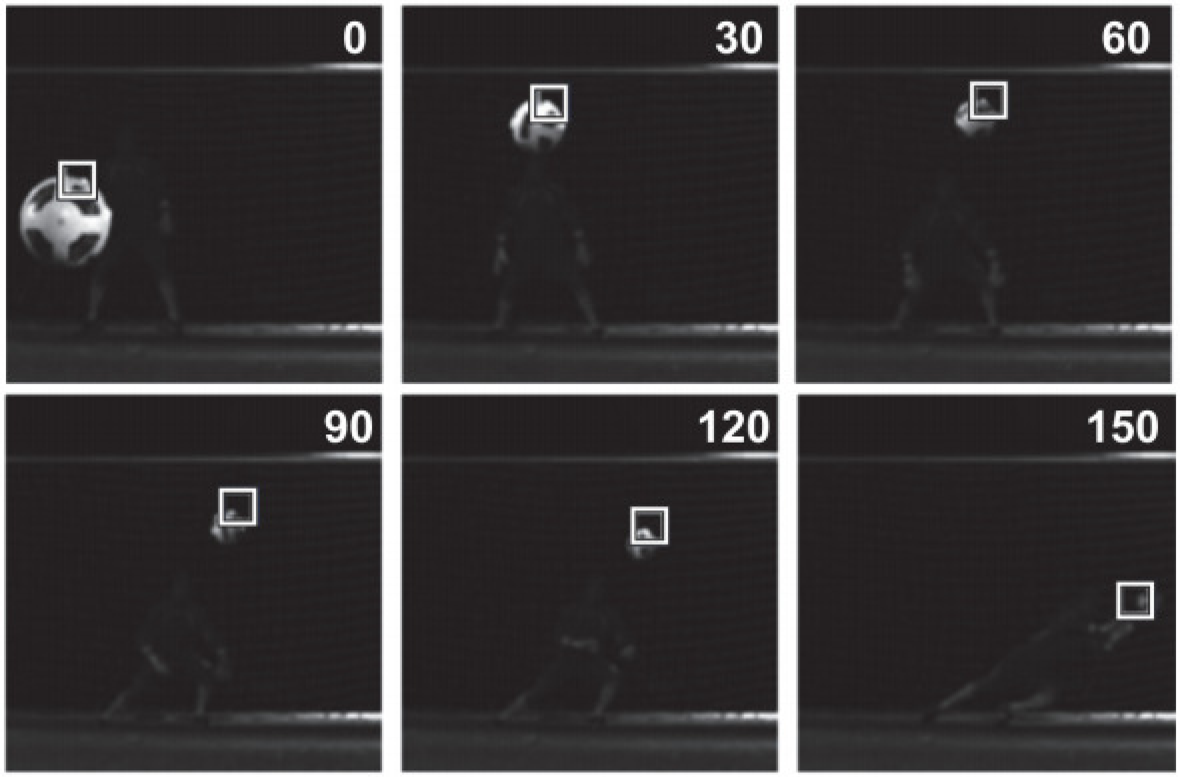


Figure 4.26. Selected frames of the network tracking a soccer ball in a slow motion video with the frame number listed in the top-right corner. The gaze generated by the tracking RBM is represented by the black and white square.

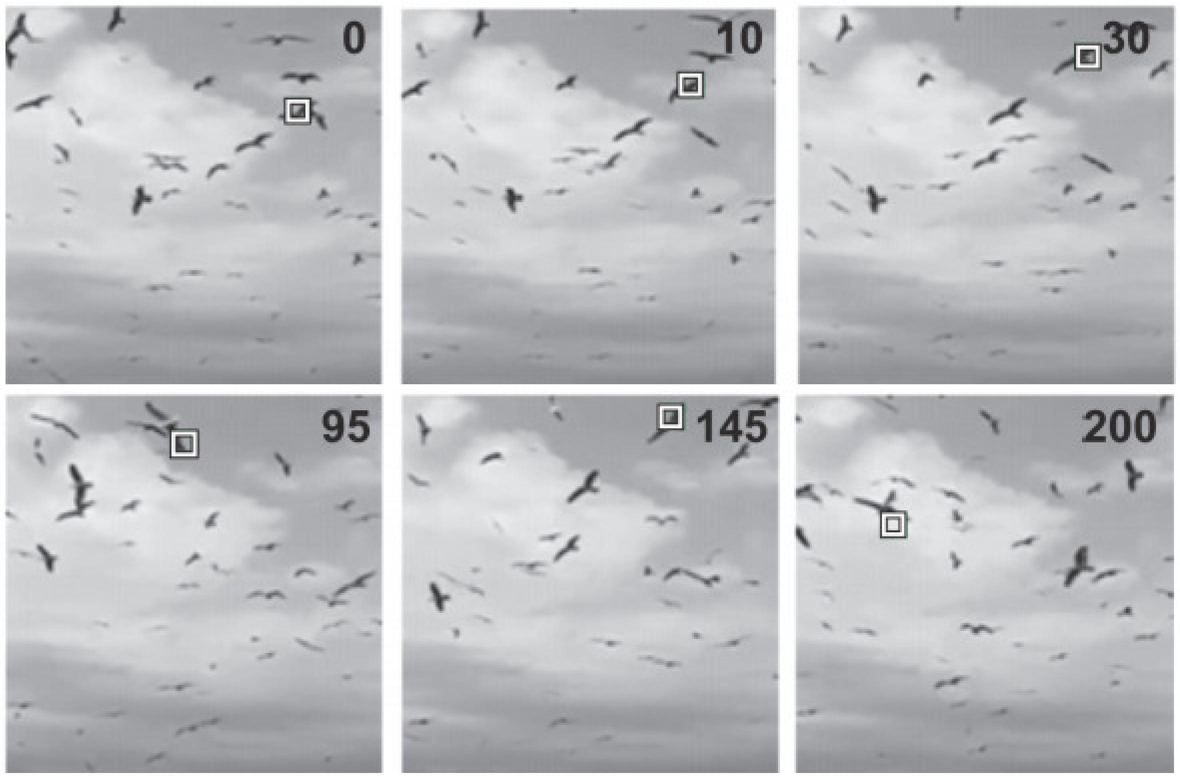


Figure 4.27. Selected frames of the network tracking a seagull in a slow motion video of a flock of seagulls with the frame number listed in the top-right corner. The gaze generated by the tracking RBM is represented by the black and white square.

Though the proposed method is meant to operate in a more realistic domain than other biological simulations, it is interesting to see how it compares to a standardized computer vision tracking algorithm. For comparison, the performance of the proposed model is compared to the performance of a template matching tracker using the squared difference comparison function on the raw grayscale frames and a sliding window search [164]. Without any additional filtering, still limiting the search space to -2 to +2 range in both axes, updating the template every frame, and using the same patch size and initial position, the competing methods are applied to different example videos.

Figure 4.28 shows both the proposed model and the template matching tracker applied to tracking a vehicle on the freeway.

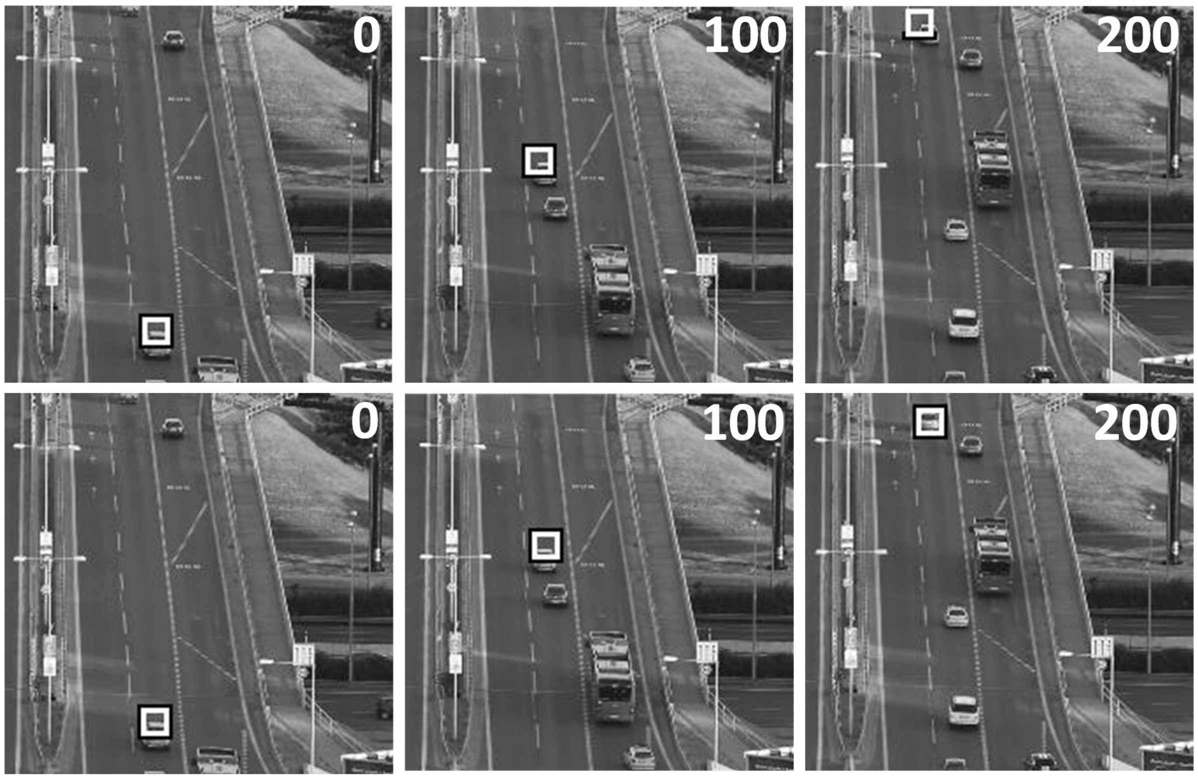


Figure 4.28. (top) Proposed model tracking a vehicle on the freeway from the bottom to the top of the frame, and (bottom) template matching tracker tracking the same vehicle.

The proposed model and the template matching tracker both track the vehicle similarly across 200 frames. The template tracker remains precise and locked onto the same portion of the car with only a slight downwards drift, while the proposed model drifts a little bit to the left of the car. The scale of the car changes slightly as it drives away, and both methods are able to tolerate that. Also, it is a very smooth linear motion, which shows that the methods perform similarly when applied to an example with simple motion, in comparison with the previous examples presented. The fact that the proposed model drifts shows that it operates differently than the template matching algorithm. Given similar patterns as the template matching algorithm, it does not necessarily produce the same result. As mentioned earlier, this is because some patterns are learned more strongly than others in the motion encoding layer and the learned correlation between motion and gaze control may not be correct due to the ambiguity and stochastic nature of the learning process in the motion tracking layer; however, for a system that learns to perform a similar task as a hardcoded template matching tracker, this result proves that it is effective, even though it is a little less accurate.

Considering a more complex scenario, Figure 4.29 shows both the proposed model and a template matching tracker applied to tracking a pedestrian at a crosswalk. This example shows a scene where the target pattern changes over time, moves in a more jittery motion, and there exists a lot of other motion in the surrounding regions.

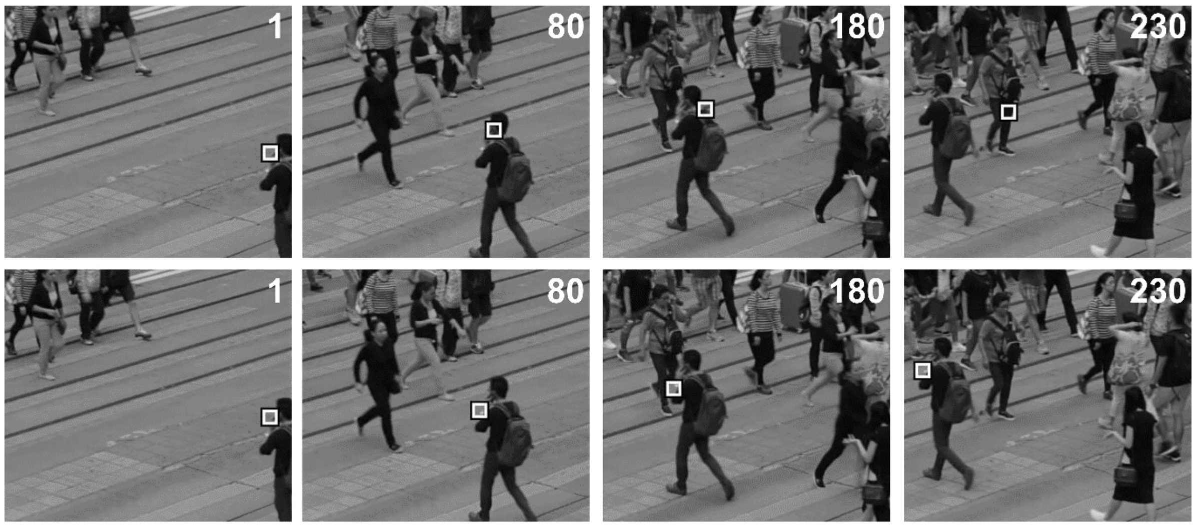


Figure 4.29. (top) Proposed model tracking a pedestrian at a crosswalk from the right of the frame to the left of the frame, and (bottom) template matching tracker tracking the same pedestrian.

It can be seen that the proposed model is able to follow the pedestrian well, despite a jittery motion and a changing pattern. However, the template matching tracker follows the exact position more closely, while the proposed model drifts to different parts of the moving pedestrian, eventually finding itself tracking the border of the person. That drift causes other motion to pass through its receptive field, thus resulting in a loss of the target before the template matching tracker does. This is a case of the difference in precision between a template matcher and the proposed model causing a significant error.

Figure 4.30 shows both the proposed model and a template matching tracker applied to tracking a surfer on the ocean.

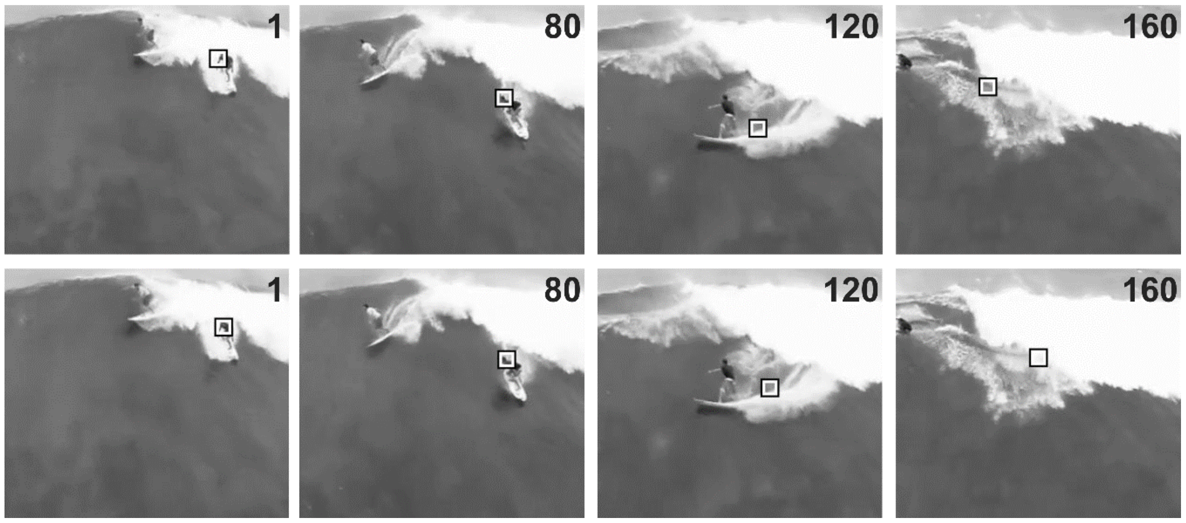


Figure 4.30. (top) Proposed model tracking a surfer on the ocean with a curved trajectory, and (bottom) template matching tracker tracking the same surfer.

The surfer video shows a more complex trajectory, moving camera, and a constantly changing target pattern. Both methods do well to track the surfer initially and show similar accuracy, however, they both fail when the surfer moves to the left of the frame. This is because the surfer travels more than 2 pixels between frames, causing both models to fail at tracking. The spatial constraint affects both methods equally, and is a problem from any motion tracking method. Many external remedies exist for this problem, such as increasing the search space or performing multiscale analysis. However, those are beyond the scope of this work, and for the goals outlined in this thesis, the proposed model matches the template tracker quite well. The main goal of this research is to improve on the reported performance of computational simulations of biological pursuit such that it can be compared in the same realm as motion tracking methods. The fact that the proposed model learns, without supervision and from real-world data only, to execute similarly to the hardcoded template matching tracker and comes close to its real-world performance makes this experiment a success relative to the objectives of this work.

It can be seen from the results that the proposed network is very effective at tracking the motion of a particular image pattern once it is in the receptive field. It is capable of tracking a wide variety of targets over a variety of motion. When the network is inaccurate, typically it is due to the motion encoder's limitations. Sometimes clutter in the scene, motion blur, or a changing target will cause the GRBM to elicit a more confused response, thus resulting in gaze movement that may not actually capture the motion. Such inaccuracies can be tolerated by using a visual attention model to reacquire the moving target. This will be demonstrated in section 4.2.10.

For short-term tracking, without using any prior information, the network has proven to be effective. It does not contain any type of temporal cohesion at this stage, and only uses the information of what is in the receptive field in subsequent frames.

4.2.9 Visual Attention

In order to demonstrate the model's performance in a visual attention task related to motion, it is also applied on real-world video based on the method described in section 4.1.7.2. The same model with the same training from section 4.2.8 is applied across the video frame to patches spaced every 10 pixels both horizontally and vertically, given frames separated by one timestep (i.e. 1 frame), and using the axis-split representation and a [-2,+2] pixel range in both axes.

Figure 4.31 and Figure 4.32 show some examples of the model finding the 5 strongest saliency locations, which correspond to pedestrians and vehicles in motion. The corresponding saliency map is also shown in those figures, where lighter colours indicate higher saliency and darker colours indicate lower saliency. This shows that the model could be used as a target selection mechanism for its own tracking, instead of using the manually-selected targets considered in section 4.2.8.

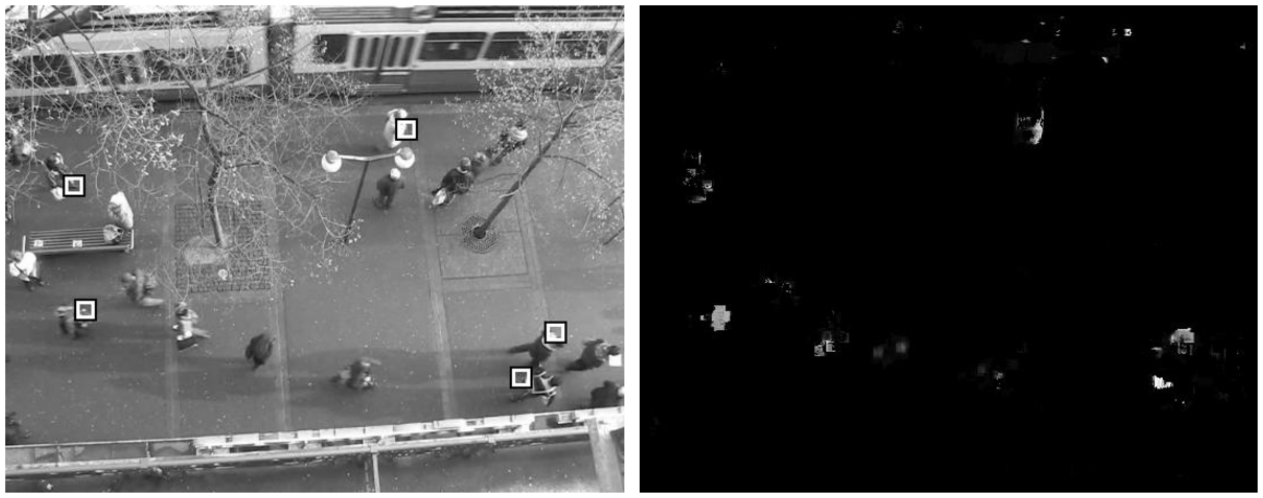

(a) (b)

Figure 4.31. (a) Visual attention selection of the 5 strongest targets denoted by the black and white squares based on (b) the motion tracking model's visual saliency calculation. Each of the targets corresponds to separate moving pedestrians.

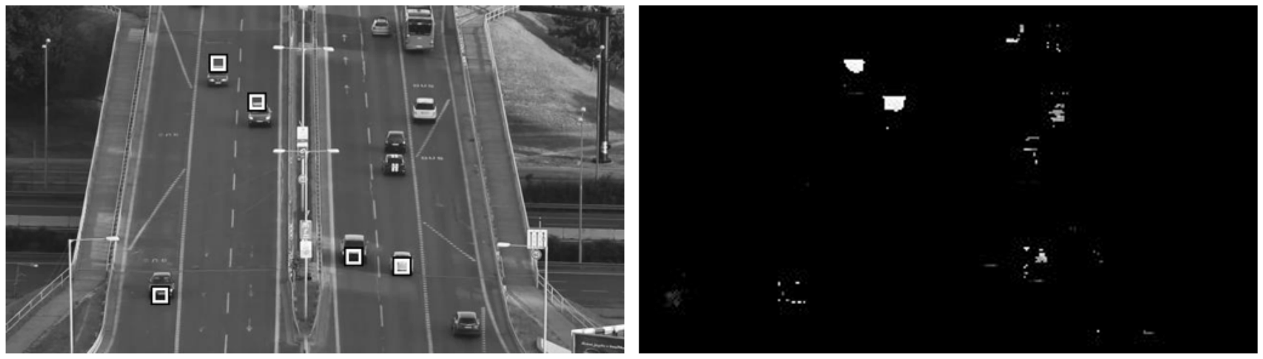

(a) (b)

Figure 4.32. (a) Visual attention selection of the 5 strongest targets denoted by the black and white squares based on (b) the motion tracking model's visual saliency calculation. Each of the targets corresponds to separate moving vehicles.

The saliency is based on the motion tracking model's computation, or what would cause the model the most excitation. This corresponds to things that move and the patterns that it has learned. This does not necessarily correspond to speed or brightness. Since all patterns and motion are not learned uniformly, as depicted by the erroneous predictions in Figure 4.18, certain areas are more salient than others. Thus, certain vehicles and pedestrians are not detected even though they are brighter, more distinct, or moving, because they don't have a pattern of interest.

Since the model does not have any object recognition capability and does not understand which targets are valuable semantically, sometimes it also finds other targets salient; these may not be useful when it comes to target selection as demonstrated in Figure 4.33, where the moving train in the background attracts the model enough for it to produce several targets there.

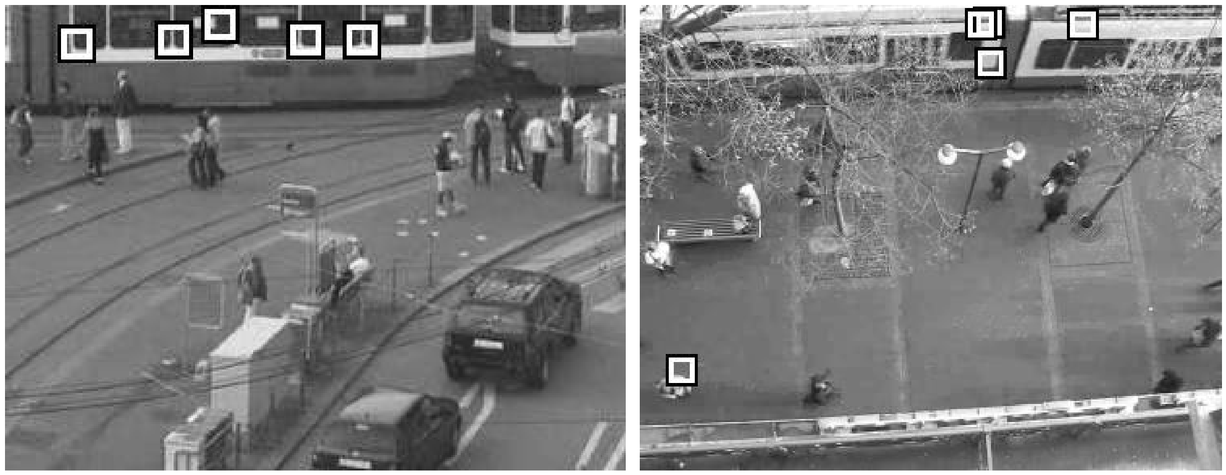


Figure 4.33. Two scenes exemplifying the motion tracking model used for visual saliency calculation, and visual attention selection of the 5 strongest targets. Many of the targets correspond to the moving trains in the background, which is not as useful for target selection.

In stationary background scenes, where foreground moving objects are the ones that are most interesting, the results show that this motion tracking model can behave as a visual attention model also.

4.2.10 Retinal Slip Compensation

As discussed in section 4.1.7.3 and 2.2.6.3, sometimes there will be drift due to accumulated errors in the tracking. The drift often results in the target moving out of the receptive field and the model staying stationary on an area of the background due to no perceived motion. By combining the motion tracking model with the visual attention implementation of that model, a more complete tracking system can be created; one that can detect a target and compensate for drift.

The same visual attention model from section 4.2.9 and the same motion tracking model from section 4.2.8 is used. The most salient target in the scene is selected for tracking, and a target is re-selected within a 50x50-pixel window centered at the center of the tracking model's 14x14-pixel receptive field when there has been no motion for 5 frames.

Figure 4.34 illustrates a target being tracked and then lost over time compared to a target being tracked without any drift due to repeated re-selection by the visual attention model. The drift here occurs due to the speed of the object exceeding the range of the motion tracking model. The results show that larger speeds can be tolerated by an increased search range with the visual attention mechanism.

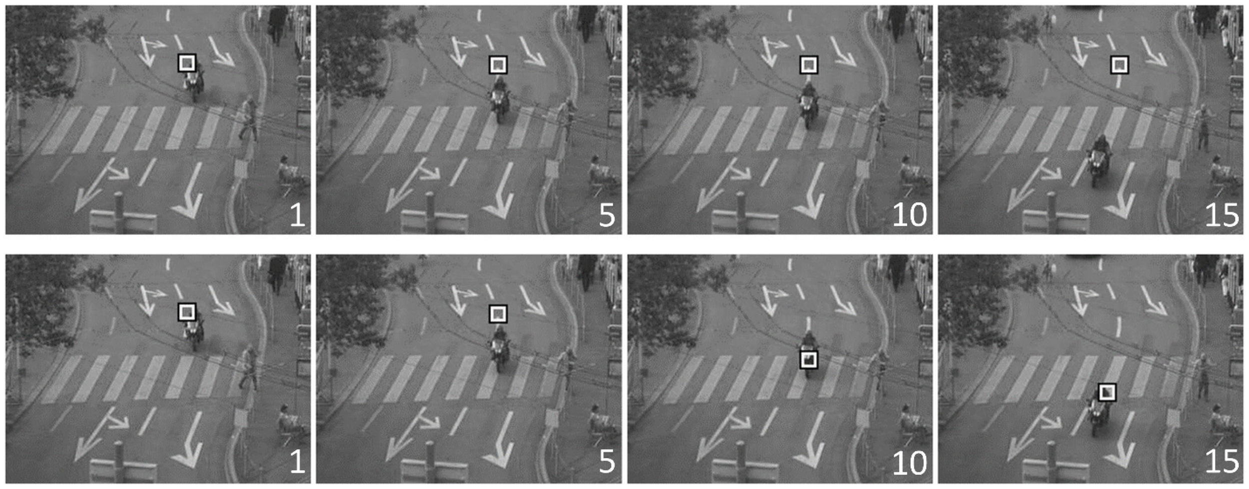


Figure 4.34. (top) Example sequence of the model losing a target due to retinal slip, and (bottom) example of the visual attention model performing retinal slip compensation on the same sequence when the target is lost in frame 5, to recover the target in frames 10 and 15.

Though it is more computationally expensive to create a saliency map than to track motion directly, it increases the robustness of the tracking and reduces the likelihood of losing the target altogether, while providing dynamic recovery means in faster or more complex video sequences.

4.2.11 Performance

To get an idea of the execution time of the above-mentioned experiments, several execution time results are presented here. The model's average training time, execution time, and saliency map execution time, accumulated over 10 trials, are displayed in Table 4.2. The experiments were performed on a machine with an Intel i7-4720HQ 2.6GHz processor and 8GB of RAM running Ubuntu 14.04.

Action	**Execution Time**
Training	
Motion encoding layer (5 million samples)	1h11m
Motion tracking layer (250000 samples)	1m44s
Execution	
Motion encoding layer	12.3ms
Motion encoding layer (pooled)	14.3ms
Motion tracking layer (1 timestep)	0.2ms
Complete tracking model (1 timestep)	14.8ms
Saliency map calculation (426 x 240 pixel image)	440.7ms

Table 4.2. Average execution times of important training and execution actions reported in the results section.

The training times are reasonable in length. The execution times are also quick, such that the model can be applied on real-time video that exceeds 60 frames a second. Expectedly, the saliency map calculation is time consuming. The search for a salient position at each frame is costly, and as a result, motion tracking as opposed to saliency-based object detection is a much faster way to follow a target from frame to frame. When necessary, during compensation for retinal slip, the computation time is slower.

4.3 Positioning the Research

4.3.1 Relationship to Existing Work

Though many of the concepts in this thesis are novel, and others have only recently been explored, it is important to compare the proposed architecture to existing work.

Conceptually, the most similar works are those regarding active inference. As mentioned in section 2.1, the active inference perspective ably links perception and action through a singular mechanism which reduces free energy, corresponding to the system reducing the likelihood of surprising sensory data. This is done through a generative model, which can represent input data and produce data of its own. Prior work in active inference uses simplistic representations, and this thesis aims to move beyond that into more complex representations. The proposed work continues to use generative models, yet does so by learning its priors as opposed to being preprogrammed with them, thus unifying learning, perception, and action.

In simpler scenarios such as visual attention based on limited priors [28] and smooth pursuit on 1D binary vectors [87], a less complex model is possible due to the simplicity of the stimuli, and learning is not required to uphold the principles of reducing free energy. The proposed framework behaves in more complex scenarios involving 2D real-world data, where no priors are given. Thus, unlike the above active inference examples, the framework in this thesis learns from visual data to create its own representation and assumptions before execution; this translates to learning its prior beliefs. Learning here is a requisite for dimensionality reduction, representation, and generalization. Considering learning mechanisms which are based on the idea of free-energy minimization, such as the RBM, this bears resemblance to the active inference methodology but extends it to learning priors.

The proposed method uses complex real-world data and can be applied as a motion detector as-is, as opposed to a biological simulations under limited sensory dimensionality and complexity provided by other methods which will be listed below. At the expense of accurate biological simulation, the proposed network becomes much more applicable. In order to accomplish execution on real-world visual data, the network must both learn representations of the real world as well as execute based on data from the real world. Where most of the reviewed self-organizing works have operated within limited scenarios, this one is able to work on data with high dimensionality in a biologically-plausible fashion. For example, Adams *et al.* [87] represent a moving target as a set bit amongst unset bits in a 1D vector and assume that the target follows a sinusoidal motion. The predictable motion simplifies the adaptation of predictive parameters. The proposed work does not make any assumptions about the type of motion that will occur, as exemplified by the complex test cases reported in Figure 4.25, Figure 4.26, and Figure 4.27, uses 2D grayscale real-world video. With no prior beliefs in the proposed system, prior beliefs to encode real-world data must be learned.

Friston *et al.* [28] require a limited set of prior beliefs to confirm based on sensory data. The technique uses saccades to gather more information driven by its preset prior beliefs. The proposed framework does not have any prior beliefs about what the target may be. Also, the proposed framework uses saccades for random traversal during training, and pursuit during execution.

The smooth pursuit method from Pack *et al.* [70] was based on the distinction between background motion and target motion, with the purpose of removing potential oscillation between perceived retinal motion caused by gaze shifts. The proposed work does not focus on relative motion and instead focuses on a singular estimated motion within the receptive field. If viewed from the perspective of Pack's work, the work in Chapter 4 of this thesis assumes that movement of the eye occurs in discrete steps and has motion detection inhibited in between those steps, thus retinal oscillation would not occur. There is biological evidence for such motion inhibition during saccadic movements [165, 166]. Also, the proposed framework learns the motion detectors in MT, and contains no predetermined rigid connections between nodes. Finally, it does not use artificial binary input movies, but uses real-world grayscale video.

The work presented by Duran *et al.* [84] assumes that the calculation between the target and the center of gaze is simple, considering that their stimuli consists of a black dot on a white background. However, real-world data makes such a process more complicated computationally since there is a great deal more ambiguity. Visually, tracking a real-world 2D moving target is complicated, and likely contains background and foreground motion. It also contains a variety of different pixel intensities. Motion can be in any direction, and though more sophisticated prediction can occur based on real-world constraints such as velocity and acceleration similarities, the proposed approach relies only on positional locality. It does not take higher-order motion into account. It only concerns itself with position and velocity estimation from adjacent frames, but does not carry over any knowledge between timesteps.

Furman and Gur [85] assume that smooth pursuit already occurs, and use their model to learn how to associate retinal reference frame information to a global reference frame. Through training, some nodes only recognize global motion, and fire despite the gaze remaining fixed or in motion as long as the object is in motion globally. The proposed framework is not concerned with the global reference frame, but provides the framework to do so. Given that the system knows the position of the virtual eyes at any given time, and the stimulus drawn is also perceived by the system, it inherently has the knowledge for that calculation and could be implemented as higher-level layers on top of the tracking layer.

Shibata *et al.* [71] uses a more traditional control system for pursuit. The innovation here is that an RNN acts as a predictor of velocity. Our model also acts as a predictor of velocity, however it generates motion independent of a control system architecture. It does not contain a previous state, does not propagate error between timesteps, and works only within the current state. Thus, unlike most machine vision tracking applications, it does not contain any filtering. The architecture is open to allowing information about previous states to improve estimations through its stacked network architecture. Since eye movements are generated by top-down information from the motion tracking layer, it is possible to have higher layers influence the motion tracking layer to integrate previous state information, such as prior velocity or acceleration.

Mahadevan and Vasconcelos [139] bias the saliency calculations for visual attention to isolate the target in each frame, provided it is within a finite search region. Wang and Yeung [140] use a neural network to distinguish between the target and the background to aid in the search. These techniques are explicitly hard-coded region-constrained search algorithms, whereas the proposed technique can be implicitly considered a region-constrained search algorithm where the possible transformations are learned via unsupervised training and the search region is the receptive field. Also, in the proposed technique, no explicit discrimination is made between the target and the background, and the prominent motion within the receptive field is taken as the target motion.

To emphasize the lack of explicit search, the proposed technique avoids search during execution by associating motion encoders to gaze movements through random saccadic search during training.

Assuming that the motion encoder comprises several motion detectors, each of which are effectively high frame rate positional change detectors, this training allows local visual search to be carried out faster. Explicit search requires more time, since it requires more complex processing, thus it is a lower frame rate detector. By leveraging high frame rate motion detectors to follow a target with best estimates, the search time can be reduced or eliminated altogether during tracking. This is the expected biological reasoning behind why pursuit is based on motion detection and not repeated search. Computationally, the speed of motion encoding may indeed be faster than explicit search, however, that is implementation-specific.

Given the literature review, this architecture is the first of its kind to use fully unsupervised learning to create motion detectors and motion tracking behaviour, and to be trained and executed on real-world video, from a biologically-inspired machine learning perspective. It is also the first of its kind to use the discriminative and generative properties of the RBM to both acquire gaze position and produce it, based on inference in a deep network.

4.3.2 Biological Parallels

In order to show that the proposed theory is appropriately parallel to biology, there must be evidence to support each of the components and concepts presented. The idea of self-organization, directional propagation, and the connectionist approach to cortical processing are all well-covered concepts and form the fundamental framework which this architecture operates in. However, other specific concepts must also be shown to have biological parallels. These concepts include the data-driven learning of motion tracking behaviour, motion discrimination, and motor control. Also, it is important to address bi-directional pathways to allow proprioception and motor control, as well as the linking of saccadic movement with smooth pursuit movement.

There is a link between saccades and pursuit. As discussed in section 2.2.6.4, saccades are thought to be useful for visual search. Early in development, movements are done exclusively by saccades. As development progresses, the eyes alternate between saccadic movement and smooth pursuit movements for tracking. This means that there is a link between using saccades and pursuit movements in tracking, and that the emergence of pursuit behaviour at later developmental stages decreases the reliance on saccades. It is plausible that the low latency pursuit behaviour can replace the high latency saccadic search-based motion information. Also, this increase in cognitive control of action over time shows that motor control can indeed be learned or, at the very least, improved. In the proposed model, the relationship that the motion tracking layer creates during training between motion encoding and gaze control via random saccadic movements forms the basis of how to replace saccadic search with motion information. Retinal constancy also falls well within this process, where a saccade and a movement driven by motion encoding will both produce a desirable predictable response on the retina, and serves as an appropriate method to link them. Finally, the ability to control the gaze emerges via association to change from random behaviour to controllable behaviour.

Up until 3.5 months of age, infants are estimated to make between 3 and 6 million eye movements [167]. Presumably, many of them are saccades and they cover the entire visual space. Also, with that many movements, many are likely to coincide with motion of varying directions and speeds. And many of those coincidences likely contain eye movements moving to match the direction and speed of the visual motion such that the same image is produced on the retina. Though it is a much rarer occurrence, among millions of eye movements, it is likely that sufficient data exists to cover each case fully and that the brain can learn from these co-occurrences. In the proposed model, random gaze movements are used to simulate the thorough coverage of the visual space, representing saccades. Between the random gaze movements and the random motion of real-world objects, some coincident motion is likely to be observed and learned from.

Biologically, the bi-directional nature of the gaze control vector is an implementation of a reflex arc, covered in section 2.2.6.2. The reflex arc both receives sensory information related to a particular action and produces an action in response. The extension of this idea in active inference [28, 65] directly supports the gaze control vector in the proposed mechanism. As discussed in the literature review, active inference proposes an alternate theory to the reflex arc where the motor control attempts to fulfill the proprioceptive error, even if that error is generated via a top-down signal. The gaze movement vector adequately models this concept, where the vector acts as a proprioceptive input describing eye movement, but also controls the eye movement to fulfill predictions made by the network. Also, the vestibulo-ocular reflex provides precedent that eye movement can belong to such a reflex arc.

Biologically, the motion encoding layer bears resemblance to V1 and V5 regions of the visual cortex. Both regions have neurons that have respond to speed and direction of moving stimuli within receptive fields. Just as theories of neural self-organization apply to the emergence of spatial features, such as those in V1 and V2, they also apply to the emergence of spatiotemporal features which can learn to encode motion, such as those in V1 and V5. In the model, the motion encoding layer is implemented as a GRBM, which learns to represent motion via spatiotemporal features. Compared to the spatial features learned in Chapter 3, this extension learns transformations, and upholds the idea of cortical self-organization.

The model's motion tracking layer combines function found in MST, V5, SC, and higher-level processing during execution. Based on the idea that pursuit behaviour is non-existent in infants and improves with age, as discussed in section 2.2.6.3, it is justifiable to believe that such behaviour is learned through exposure to real-world data and self-organization. Thus, the training of the motion tracking layer reasonably represents that idea, by being trained with real-world data and self-organizing to carry out pursuit behaviour.

Higher-level cognition in the brain dictates whether a target is tracked or not. In the execution of the model, the retinal constancy value is set to 1 such that the motion tracking layer produces the right gaze movement to match the motion encoding. If the value is set to 0, the produced gaze movement is random since no discernible pattern emerges when retinal constancy is low during training. This behaviour corresponds to how higher-level cognition can enable or disable motion tracking.

Biologically, the eye movement process contains propagation of information through the pons, the cerebellum, and ultimately the optic motor neurons to control the eyes. This is a complex process to correct for errors and redirect signals through different pathways, and as a result, it is more complex than necessary for the simulation in this thesis. In this model, the actual movement of the gaze is a simplification of the eye movement process, where the signal produced intends to move the eye at a certain speed and direction and the eye performs this movement. However, the temporal lag in processing and eye movement is acknowledged by the constant gaze delay, Δt_g.

4.4 Summary

The proposed model in this chapter was described as a neural alternative to traditional motion tracking, a detailed explanation expanding novel concepts in the architecture, a unique application of existing machine learning methods, was positioned biologically, and compared to other computational methods of a similar topic found in literature.

Its originality and efficacy as compared to other work was described thoroughly in section 4.3.1, and its relationship to biology was explained in section 4.3.2.

There are several novel concepts within the proposed method which are re-summarized below:

- A simple deep network architecture for motion tracking similar to that found in biology;
- Retinal constancy as a single value to drive the self-organization of a network. Motion tracking behaviour emerges by using retinal constancy as a link between random motion and random gaze movements;
- Using unsupervised learning for both motion encoding and motion tracking;
- Exploiting the generative properties of an RBM to learn action as well as produce action. This corresponds to the unsupervised learning of a control system with the same mechanism used to learn representation;
- Showing that real-world data for training is sufficient to train the network well enough to execute on real-world data; It is important to note the generalized behaviour of the network. In the experimental results, it is not trained on the same data that it is tested on, nor does it need to be. It can be trained with synthetic motion and applied to real-world motion, vice versa, and using different datasets or different parts of the same dataset.
- Biologically linking saccades to pursuit behaviour, as an explanation for the improvement of pursuit in early development. The literature shows that as smooth pursuit improves it becomes less reliant on saccadic jumps. However, the proposed model uses saccades as a reason for the improvement;
- Using importance weighting when training an autoassociative network, as described in section 4.1.5.
- The use of the same motion tracking model generalized for visual saliency, and its application to compensate for retinal slip (or drift), as described in 4.1.7. This improvement makes the model more robust to realistic motion tracking scenarios when compared to other biologically-inspired models of this nature.

As a whole, the concept of motion tracking is well-researched in machine vision, and its counterpart, smooth pursuit, is well-researched in biology. There is less focus on computational models of smooth pursuit, and most of the emphasis in the literature is found in specific implementations to simulate certain aspects of it. Where most machine learning focuses on representation learning, this part of the thesis aims to promote the idea of unifying perception and action through the representational and generative capabilities of a generative model. Though the architecture is simple, the idea is complex, and its efficacy is a testament of the practicality in this line of thinking. Finally, its results show that it performs well on real-world data both for training and execution, which is a significant leap forward from such simulations that are typically performed on synthetic data. The real-world tests show that the proposed model is not far from the traditional machine vision template matching technique in terms of effectiveness even though it has to learn its behaviour and constraints, which already exceeds the boundaries within which most biological simulations are based. The proposed motion tracking model is positioned well within the realm of computational models of biological smooth pursuit, machine vision, and unsupervised learning, and serves to contribute novel ideas to each of those fields. This part of the research has been so far published in [168] and submitted to [169].

Chapter 5 Conclusion

5.1 Summary

This thesis covered two major topics of research to self-organizing neural visual models for applications in machine vision. The first topic is the evaluation of different methods of biasing RBMs with local neighbouring information, and quantifying their feature extraction effectiveness in the visual domain. It has long been thought that forcing independence upon units in unsupervised learning leads to better features; such as ICA, K-Means, and RBMs. However, the work presented in chapter 3 challenges that notion, and does so by incorporating local information to improve the features developed during learning from the regular case. The efforts are proven to regularly improve performance in classification on visual datasets in different domains. Second, a self-organizing, biologically-inspired motion tracking model is developed in chapter 4, which represents the main contribution of this thesis. Starting from a two-layer deep network, in an uninitialized state, presenting real-world video and allowing control over its gaze, allows the network to eventually learn to control its gaze to follow detected visual motion. As a method of reinforcement learning, this technique proves effective on real-world datasets and provides a theory as to how smooth pursuit behaviour can develop from just exposure to real-world video and a simple learning mechanism. A number of original contributions were made in this thesis, and they are summarized in section 5.2, along with a list of publications that resulted from this research. Finally, a brief overview of remaining challenges that can be addressed in future work to further develop both of the main components of this project are outlined section 5.3.

5.2 Original Contributions

In this thesis, several original contributions were made. They are divided into the two main topics covered by this thesis, listed in reverse order here to better emphasize the more significant contributions.

5.2.1 A Biologically-Inspired Self-Organizing Motion Tracking Model

The principal contribution of this thesis is the motion tracking model which self-organizes via fully unsupervised learning from real-world video data, discussed in Chapter 4. The model can be broken down into several original concepts.

- The model itself is the first contribution, described in section 4.1. To the best of my knowledge, it is the first model to use unsupervised methods to learn visual motion encoding, tracking behaviour, and the association between the two.
    - In motion tracking methods, this association is often hardcoded such as those discussed in section 2.2.6.3 and emphasized in section 4.3.1. However, keeping in line with the fully unsupervised nature of the system, an important contribution is that this property is also learned.
- The second contribution is a novel theory for the emergence of smooth pursuit behaviour rooted in the physiology of the visual system involving saccades, visual expectation, and motion sensitivity during early development in the visual cortex. This is accomplished through the novel retinal constancy concept, which is only a single real value that guides the self-organization of a

system that operates in a high-dimensional space, resulting in the emergence of 2D action generation and 2D motion perception.

  - This value can be considered akin to reward in reinforcement learning or fitness in evolutionary computation. It provides no 2D information to guide the network towards learning preferences, and only produces a single value which determines if there is congruency between perception and action. A more detailed description of retinal constancy was presented in section 4.1.5, while the biological inspiration for this theory was presented in section 4.3.2. The use of generalized unsupervised machine learning algorithms emphasizes the robustness of the theory as opposed to application-specific implementation details such as the other computational simulations of biological pursuit discussed in section 2.2.6.3 and 2.2.7.

- The third contribution is proving a self-organizing model using complex real-world data as opposed to simple synthetic data.
  - Unlike most computational simulations of biological pursuit, as discussed in sections 2.2.6.3, 2.2.7, and 4.3.1, the proposed system uses real-world data for both training and execution. This thesis places the focus on the novelty, biological parallels, simplicity, and efficacy of biologically-inspired methods. The intended goal is not to match motion tracking performance of the most proven machine vision methods, but to exceed the performance of computational simulations of biological pursuit to bridge the gap between machine learning results and biological simulation results. In that, self-organization from only real-world data makes a major difference.
- The fourth contribution is showing the benefit of using analog nodes in the RBM to interpolate underrepresented patterns in the training data for analog control by a generative model, described in section 4.1.4.3 and tested in section 4.2.2.
  - Using a generative model, learned in an unsupervised manner, as a control system is already a rare concept. Therefore, this thesis contributes an original model to that field, and the use of analog control through real-valued input nodes in the RBM is an additional contribution beyond that. Currently, its performance does not surpass the other methods of gaze representation, however, further work could improve this concept to become competitive with one-hot representations.
- The fifth contribution is the spatial filtering of motion using convolution to increase the stability of the motion encoding.
  - By increasing the size of the receptive field, noise and inconsistencies can be reduced by the motion encoding GRBM to produce a good estimation of the motion. However, the exponential increase in parameters required to increase the receptive field size makes it an infeasible solution. The convolution procedure detailed in 4.2.1.1 allows a smaller model to be applied simultaneously on the target patch as well as the surrounding patches such that the results can be averaged to provide a more stable signal. This also indirectly increases the receptive field size without the parameter explosion.
- The sixth and final original contribution is approaching the creation of a motion tracker from the perspective of minimizing designer influence, reducing the number of assumptions made, and learning directly from the data instead of hardcoding behaviour.

5.2.2 Biasing Unsupervised Single-Layer Networks using Gaussian Filters to Learn Invariant Visual Features

Finally, in the context of unsupervised visual representation learning of invariant features, discussed in Chapter 3, the main contributions were a comparison of several techniques which integrate local information both spatially, temporally, and in feature space, and challenging the notion that independently learned feature detectors are more powerful than ones incorporating neighbouring information.

- The first contribution is the measurement and comparison of discriminative abilities of the features generated by the above-mentioned techniques.
- The second contribution is the measurement and comparison of the invariant properties of the features generated by the above-mentioned techniques.

5.2.3 Publications

The following publications resulted from the research conducted for this thesis:

- [154] – related to chapter 3
  A. Yogeswaran and P. Payeur, "Improving Visual Feature Representations by Biasing Restricted Boltzmann Machines with Gaussian Filters," in *Advances in Visual Computing: 12th International Symposium, ISVC 2016, Las Vegas, NV, USA, December 12-14, 2016, Proceedings, Part I*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, F. Porikli, S. Skaff*, et al.*, Eds., ed Cham: Springer International Publishing, 2016, pp. 825-835.

- [155] – related to chapter 3
  A. Yogeswaran and P. Payeur, "Biasing restricted Boltzmann machines using Gaussian filters to learn invariant visual features," *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*, 2016, pp. 1-8.

- [168] – related to chapter 4
  A. Yogeswaran and P. Payeur, "Leveraging Saccades to Learn Smooth Pursuit: A Self-Organizing Motion Tracking Model Using Restricted Boltzmann Machines," *Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, San Francisco, California, USA, 2017, pp. 4313-4319.

- [169] – related to chapter 4
  A. Yogeswaran and P. Payeur, "Self-Organizing Smooth Pursuit Model using Retinal Constancy", submitted to Neural Computation, 2017.

5.3 Future Work

To further explore the frameworks described in this thesis, there are several additional steps that can be considered as future work.

5.3.1 A Biologically-Inspired Self-Organizing Motion Tracking Model

- *Improving spatial constraints.*
    - There are several spatial constraints present in the current state of this motion tracking model. The first is the receptive field window, which is kept at a modest 14x14 size. This can be remedied with a larger visible layer in the GRBM, but to learn a good model of transformations in a larger visible layer, it requires a larger number of factors and hidden nodes. As a result, a larger network as a whole must be created and trained. It is only practical to do this with larger resources, such as a GPU cluster.
    - The second spatial constraint is the motion range, which is kept at a maximum of [-2, +2] pixels in both axes. This can be increased directly by increasing the gaze movement vector size. However, to properly encode the correlations, the motion tracking hidden layer size must also increase. Again, the larger the range gets, the larger the network must get. To increase the size to a much more practical motion range, a larger set of resources for training and execution would be required. Again, a GPU cluster would be suitable to test this network at a larger scale.
- *Improving temporal constraints.* The model also contains temporal constraints. It is only able to recognize correlations between two video frames. Ideally, it would be able to acquire information from a series of timesteps to make its predictions. Using a temporal RBM, such as the one described in [170], would bring a temporal solution to a problem in the temporal domain, making this a more realistic approach. It will allow correlations across larger temporal ranges, and would not require manual interpolation or excess discretization of temporal data. However, temporal RBMs are known to be volatile and difficult to train.
- *Analog control.* The analog gaze movement vector representation is a potentially powerful form of gaze control for this application. It is simple and biologically-plausible to use a single node to control gaze movement in one axis, similar to a single muscle group expanding or contracting to pull the eye in one direction. This thesis has shown that it can work, however, its performance is still not as strong as the other gaze movement vector representations. By altering the motion tracking layer RBM to be better suited to using real-valued data, the analog gaze movement vector representation would be much more effective than it is in the current model.
- *Reinforcement Learning.* Framing the topic in the domain of reinforcement learning would bridge the gap between our independent work and state-of-the-art research in literature. It would also open up the work to a rigid formulation and allow a wider variety of algorithmic tools to solve this problem, being in line with other work such as [26] and [27].
- *Using a Top Layer.*
    - Keeping with the concept of biological inspiration, a valuable idea would be to use a layer above the motion tracking layer to produce top-down influences and act as a Kalman filter, allowing information from previous timesteps to drive future movements. This will help with prediction and anticipatory movements.

    - In the same vein, using a top layer to mimic open-loop and closed-loop pursuit types based on prior knowledge would bring this model closer to a complete pursuit system.
- *Colour.* The use of colour data instead of grayscale data would improve tracking accuracy, since the extra information would disambiguate similar transformations which confuse the network. Naturally, using the extra information would also require an increase in network size; it will definitely require an increase in visible nodes in the motion encoding layer to handle the multi-channel representation, and may require an increase in other parameters to keep the discriminative ability high. It would also improve tracking accuracy in the motion tracking layer, since the retinal constancy calculation would disambiguate the same confusing situations found in the motion encoding layer.

5.3.2 Biasing Unsupervised Single-Layer Networks using Gaussian Filters to Learn Invariant Visual Features

- Currently, only a single layer network is used. However, to see if the feature learning performance improves as the network becomes multi-layered, the methodologies should be applied on a second layer. Concatenating the outputs of those layers will give a richer representation; this representation can be tested by using the concatenated feature vector as the input to the classifier to see if there is a fixed improvement as there was with just a single layer.
- Pooling is commonly used to induce invariance in classification models. It would be valuable to determine how pooling affects invariant feature representations versus standard feature representations, such as the ones compared in this thesis.
- Apply these training methodologies to existing state-of-the-art convolutional networks would help determine if these methods can affect classification performance at the highest accuracies found on more complex datasets, such as [116]. These methodologies would be used to initialize them prior to backpropagation training. It would be interesting to see if there is a comparable performance increase, or if the back propagation negates the benefits of these methodologies.
- Currently, only 3x3 kernels are used for the discrete Gaussian filter, but larger kernel sizes would help determine feature improvements relative to a different parameter. Though unlikely that the performance would increase much due to a larger kernel, at least not enough to offset the computational complexity, it would help further characterize the role of information sharing between neighbouring nodes.

References

[1] J. Xu, G. Chen, and M. Xie, "Vision-guided automatic parking for smart car," *Proceedings of the IEEE Intelligent Vehicles Symposium*, Dearborn, Michigan, USA, 2000, pp. 725-730.

[2] E. Dagan, O. Mano, G. P. Stein, and A. Shashua, "Forward collision warning with a single camera," *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2004, pp. 37-42.

[3] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE Transactions on Intelligent Transportation Systems,* vol. 4, 2003, pp. 143-153.

[4] E. Guizzo, "How Google's self-driving car works," *IEEE Spectrum Online, October,* vol. 18, 2011.

[5] X. Zhang and Y. Gao, "Face recognition across pose: A review," *Pattern Recognition,* vol. 42, 2009, pp. 2876-2896.

[6] J. Z. Leibo, J. Mutsch, and T. Poggio, "Why the brain separates face recognition from object recognition," *Advances in Neural Information Processing Systems*, 2011, pp. 711-719.

[7] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 34, 2012, pp. 743-761.

[8] M. Bar, "Visual objects in context," *Nature Reviews Neuroscience,* vol. 5, 2004, pp. 617-629.

[9] T. Brosnan and D.-W. Sun, "Improving quality inspection of food products by computer vision - a review," *Journal of Food Engineering,* vol. 61, 2004, pp. 3-16.

[10] Y.-R. Chen, K. Chao, and M. S. Kim, "Machine vision technology for agricultural applications," *Computers and Electronics in Agriculture,* vol. 36, 2002, pp. 173-191.

[11] K. Kapach, E. Barnea, R. Marion, Y. Edan, and O. Ben-Shahar, "Computer vision for fruit harvesting robots-state of the art and challenges ahead," *International Journal of Computational Vision and Robotics,* vol. 3, 2012, pp. 4-34.

[12] D. W. F. Van Krevelen and R. Poelman, "A survey of augmented reality technologies, applications and limitations," *International Journal of Virtual Reality,* vol. 9, 2010, p. 1.

[13] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation,* vol. 18, 2006, pp. 1527-1554.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature,* vol. 521, 2015, pp. 436-444.

[15] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, and J. Williams, "Recent advances in deep learning for speech research at Microsoft," *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 8604-8608.

[16] T. K. Hensch, "Critical period regulation," *Annual Review Neuroscience,* vol. 27, 2004, pp. 549-579.

[17] N. W. Daw, "Mechanisms of plasticity in the visual cortex," *Investigative Opthalmology and Visual Science,* vol. 35, 1994, pp. 4168-4179.

[18] T. Wiesel and D. Hubel, "Single-cell responses in striate cortex of kittens deprived of vision in one eye.," *Journal of Neurophysiology,* vol. 26, 1963, pp. 1003-1017.

[19] T. Poggio and S. Ullman, "Vision: Are models of object recognition catching up with the brain?," *Annals of the New York Academy of Sciences,* vol. 1305, 2013, pp. 72-82.

[20] F. Mushtaq, A. R. Bland, and A. Schaefer, "Uncertainty and cognitive control," *Frontiers in psychology,* vol. 2, 2011.

[21] G. Hinton, L. Deng, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine,* vol. 29, 2012, pp. 82-97.

[22] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng, "Building High-Level Features with Large Scale Unsupervised Learning," *International Conference on Machine Learning (ICML '12)*, Edinburgh, Scotland, 2012, pp. 81-88.

[23] A. Coates, H. Lee, and A. Ng, "An Analysis of Single-Layer Networks in Unsupervised Feature Learning," *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 215-223.

[24] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction* vol. 1: MIT press Cambridge, 1998.

[25] R. Samson, M. Frank, and J.-M. Fellous, "Computational models of reinforcement learning: the role of dopamine as a reward signal," *Cognitive neurodynamics,* vol. 4, 2010, pp. 91-105.

[26] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot, "Mastering the game of Go with deep neural networks and tree search," *Nature,* vol. 529, 2016, pp. 484-489.

[27] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *International Conference on Machine Learning*, 2015, pp. 2048-2057.

[28] K. Friston, R. A. Adams, L. Perrinet, and M. Breakspear, "Perceptions as Hypotheses: Saccades as Experiments," *Frontiers in Psychology,* vol. 3, 2012.

[29] S. K. M. Yi, M. Steyvers, M. D. Lee, and M. J. Dry, "The wisdom of the crowd in combinatorial problems," *Cognitive Science,* vol. 36, 2012, pp. 452-470.

[30] S. Murata and H. Kurokawa, *Self-organizing robots* vol. 77: Springer, 2012.

[31] J. S. Kelso, *The self-organization of brain and behavior*: MIT press, 1997.

[32] D. O. Hebb, *The Organization of Behavior*. New York: Wiley & Sons, 1949.

[33] D. C. Knill and A. Pouget, "The Bayesian brain: the role of uncertainty in neural coding and computation," *TRENDS in Neurosciences,* vol. 27, 2004, pp. 712-719.

[34] K. Friston and K. E. Stephan, "Free energy and the brain," *Synthese,* vol. 159, 2007, pp. 417-458.

[35] K. Friston, "The free-energy principle: a unified brain theory?," *Nature Reviews Neuroscience,* vol. 11, 2010, pp. 127-138.

[36] S. Kullback, *Information Theory and Statistics*: John Wiley & Sons, 1959.

[37] G. Hinton, P. Dayan, B. Frey, and R. Neal, "The wake–sleep algorithm for unsupervised neural networks," *Science,* vol. 268, 1995, pp. 1158-1161.

[38] K. Friston, "Learning and inference in the brain," *Neural Networks,* vol. 16, 2003, pp. 1325-1352.

[39] K. Friston, "What is optimal about motor control?," *Neuron,* vol. 72, 2011, pp. 488-498.

[40] S. Zeki, *A Vision of the Brain*: Blackwell scientific publications, 1993.

[41] N. K. Logothetis, J. Pauls, and T. Poggio, "Shape representation in the inferior temporal cortex of monkeys," *Current Biology,* vol. 5, 1995, pp. 552-563.

[42] S. Marcelja, "Mathematical description of the responses of simple cortical cells," *Journal of the Optical Society of America,* vol. 70, 1980, pp. 1297-1300.

[43] A. Anzai, X. Peng, and D. C. Van Essen, "Neurons in monkey visual area V2 encode combinations of orientations," *Nature neuroscience,* vol. 10, 2007, pp. 1313-1321.

[44] M. F. López-Aranda, J. F. López-Téllez, I. Navarro-Lobato, M. Masmudi-Martín, A. Gutiérrez, and Z. U. Khan, "Role of layer 6 of V2 visual cortex in object-recognition memory," *Science,* vol. 325, 2009, pp. 87-89.

[45] S. J. Luck, L. Chelazzi, S. A. Hillyard, and R. Desimone, "Neural mechanisms of spatial selective attention in areas V1, V2, and V4 of macaque visual cortex," *Journal of neurophysiology,* vol. 77, 1997, pp. 24-42.

[46] C. D. Salzman, K. H. Britten, and W. T. Newsome, "Cortical microstimulation influences perceptual judgements of motion direction," *Nature,* vol. 346, 1990, p. 174.

[47] B. Fischer and B. Breitmeyer, "Mechanisms of visual attention revealed by saccadic eye movements," *Neuropsychologia,* vol. 25, 1987, pp. 73-83.

[48] C. J. Bruce, M. E. Goldberg, M. C. Bushnell, and G. B. Stanton, "Primate frontal eye fields. II. Physiological and anatomical correlates of electrically evoked eye movements," *Journal of neurophysiology,* vol. 54, 1985, pp. 714-734.

[49] M. A. Sommer and R. H. Wurtz, "Composition and topographic organization of signals sent from the frontal eye field to the superior colliculus," *Journal of Neurophysiology,* vol. 83, 2000, pp. 1979-2001.

[50] S. E. Petersen, D. L. Robinson, and J. D. Morris, "Contributions of the pulvinar to visual spatial attention," *Neuropsychologia,* vol. 25, 1987, pp. 97-105.

[51] D. L. Robinson and J. W. McClurkin, "The visual superior colliculus and pulvinar," *Reviews of oculomotor research,* vol. 3, 1988, pp. 337-360.

[52] D. J. Jobson, Z.-u. Rahman, and G. A. Woodell, "Properties and performance of a center/surround retinex," *IEEE Transactions on Image Processing,* vol. 6, 1997, pp. 451-462.

[53] A. Hyvärinen and E. Oja, "Independent Component Analysis: Algorithms and Applications," *Neural Networks,* vol. 13, 2000, pp. 411-430.

[54] R. T. Born and R. B. H. Tootell, "Segregation of global and local motion processing in primate middle temporal visual area," *Nature,* vol. 357, 1992, pp. 497-499.

[55] D. Hubel and T. Wiesel, "Receptive Fields and Functional Architecture of Monkey Striate Cortex," *J. Physiol,* vol. 195, 1968, pp. 215-243.

[56] N. Ramalingam, J. McManus, W. Li, and C. Gilbert, "Top-Down Modulation of Lateral Interactions in Visual Cortex," *The Journal of Neuroscience,* vol. 33, 2013, pp. 1773-1789.

[57] W. Reichardt, "Evaluation of optical motion information by movement detectors," *Journal of Comparative Physiology A,* vol. 161, 1987, pp. 533-547.

[58] J. P. H. van Santen and G. Sperling, "Elaborated Reichardt Detectors," *J. Opt. Soc. Am. A,* vol. 2, 1985, pp. 300-321.

[59] J. Wattam-Bell, "Visual motion processing in one-month-old infants: Habituation experiments," *Vision Research,* vol. 36, 1996, pp. 1679-1685.

[60] G. Gredebäck, H. Örnkloo, and C. von Hofsten, "The development of reactive saccade latencies," *Experimental Brain Research,* vol. 173, 2006, pp. 159-164.

[61] R. Memisevic and G. Hinton, "Unsupervised learning of image transformations," *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR '07)*, Minneapolis, Minnesota, USA, 2007, pp. 1-8.

[62] K. Donner and S. Hemila, "Modelling the effects of microsaccades on primate retinal responses indicated that microsaccades may enhance sensitivity to edges and improve spatial resolution," *Vision Research,* vol. 47, 2007, pp. 1166-1177.

[63] M. B. McCamy, J. Otero-Millan, S. L. Macknik, Y. Yang, X. G. Troncoso, S. M. Baer, S. M. Crook, and S. Martinez-Conde, "Microsaccadic efficacy and contribution to foveal and peripheral vision," *Journal of Neuroscience,* vol. 32, 2012, pp. 9194-9204.

[64] M. Dieterich and T. Brandt, "Vestibulo-ocular reflex," *Current Opinion in Neurobiology,* vol. 8, 1995, pp. 83-88.

[65] R. A. Adams, S. Shipp, and K. Friston, "Predictions not commands: active inference in the motor system," *Brain Structure & Function,* vol. 218, 2013, pp. 611-643.

[66] P. Thier and U. J. Ilg, "The neural basis of smooth-pursuit eye movements," *Current Opinion in Neurobiology,* vol. 15, 2005, pp. 645-652.

[67] R. N. Aslin, "Development of smooth pursuit in human infants," *Cognition and Visual Perception*, 1981, pp. 31-51.

[68] J. E. Richards and F. B. Holley, "Infant attention and the development of smooth pursuit tracking," *Developmental Psychology,* vol. 35, 1999, pp. 856-867.

[69] C. Von Hofsten and K. Rosander, "Development of Smooth Pursuit Tracking Young Infants," *Vision Research,* vol. 37, 1997, pp. 1799-1810.

[70] C. Pack, S. Grossberg, and E. Mingolla, "A neural model of smooth pursuit control and motion perception by cortical area MST," *Journal of Cognitive Neuroscience,* vol. 13, 1999, pp. 102-120.

[71] T. Shibata, H. Tabata, S. Schaal, and M. Kawato, "A model of smooth pursuit in primates based on learning the target dynamics," *Neural Networks,* vol. 18, 2005, pp. 213-224.

[72] M. C. Dorris and D. P. Munoz, "A neural correlate for the gap effect on saccadic reaction times in monkeys," *Journal of Neurophysiology,* vol. 73, 1995, pp. 2558-2562.

[73] R. N. Aslin, *Anatomical constraints on oculomotor development: implications for infant perception*. Hillsdale, NJ, 1988.

[74] R. L. Canfield and N. Z. Kirkham, "Infant cortical development and the prospective control of saccadic eye movements," *Infancy,* vol. 2, 2001, pp. 197-211.

[75] W. Wang, C. Chen, Y. Wang, T. Jiang, F. Fang, and Y. Yao, "Simulating human saccadic scanpaths on natural images," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*, Colorado Springs, USA, 2011, pp. 441-448.

[76] C. Ekanadham and H. Lee, "Sparse deep belief net models for visual area V2," *Undergraduate Thesis*, Computer Science, Stanford University, 2007.

[77] H. Lee, C. Ekanadham, and A. Ng, "Sparse Deep Belief Net Model for Visual Area V2," *Advances in Neural Information Processing Systems (NIPS)*, 2008, pp. 873-880.

[78] A. Hyvärinen and P. Hoyer, "Emergence of Phase- and Shift-Invariant Features by Decomposition of Natural Images into Independent Feature Subspaces," *Neural Computation,* vol. 12, 2000, pp. 1705-1720.

[79] A. Hyvärinen, P. Hoyer, and M. Inki, "Topographic Independent Component Analysis," *Neural Computation,* vol. 13, 2001, pp. 1527-1558.

[80] D. H. Hubel and T. Wiesel, "Sequence regularity and geometry of orientation columns in the monkey striate cortex," *Journal of Comparative Neurology,* vol. 158, 1974, pp. 267-293.

[81] R. Memisevic and G. Hinton, "Learning to represent spatial transformations with factored higher-order Boltzmann machines," *Neural Computation,* vol. 22, 2010.

[82] J. Lücke, "A dynamical model for receptive self-organization in V1 cortical columns," *Proceedings of the 17th International Conference on Artificial Neural Networks (ICANN 2007)*, Porto, Portugal, 2007, pp. 389-398.

[83] J. A. Bednar and R. Miikkulainen, "Self-organization of spatiotemporal receptive fields and laterally connected direction and orientation maps," *Neurocomputing,* vol. 52, 2003, pp. 473-480.

[84] B. Duran, G. Sandini, and G. Metta, "Emergence of smooth pursuit using chaos," *First International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2007)*, Boston, USA, 2007, pp. 269-272.

[85] M. Furman and M. Gur, "Self-organizing neural network model of motion processing in the visual cortex during smooth pursuit," *Vision Research,* vol. 2003, 2003, pp. 2155-2171.

[86] K. Friston, "Embodied inference and spatial cognition," *Cognitive Processing,* vol. 13, 2012, pp. 171-177.

[87] R. A. Adams, L. Perrinet, and K. Friston, "Smooth pursuit and visual occlusion: active inference and oculomotor contorl in schizophrenia," *PLoS ONE*, 2012.

[88] Y. Chauvin and D. E. Rumelhart, *Backpropagation: theory, architectures, and applications*: Psychology Press, 1995.

[89] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters,* vol. 9, 1999, pp. 293-300.

[90] K. P. Murphy, *Machine learning: a probabilistic perspective*: MIT press, 2012.

[91] M. E. Celebi and K. Aydin, *Unsupervised Learning Algorithms*: Springer, 2016.

[92] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability,* vol. 1, 1967, pp. 281-297.

[93] T. Kohonen, *Self-organizing maps* vol. 30, 2001.

[94] P. Smolensky, "Information Processing in Dynamical Systems: Foundations of Harmony Theory," in *Parallel Distributed Processing*. vol. 1, D. Rumelhart and J. McClelland, Eds., ed: MIT Press, 1986, pp. 194-281.

[95] G. Hinton, "Training Products of Experts by Minimizing Contrastive Divergence," *Neural Computation,* vol. 14, 2002, pp. 1771-1800.

[96] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 6, 1984, pp. 721-741.

[97] T. Tieleman, "Training restricted Boltzmann machines using approximations to the likelihood gradient," *25th International Conference on Machine Learning*, 2008, pp. 1064-1071.

[98] V. Nair and G. Hinton, "3-d Object Recognition with Deep Belief Nets," *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 1339-1347.

[99] H. Goh, M. Cord, and N. Thome, "Biasing restricted Boltzmann machines to manipulate latent selectivity and sparsity," *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2010*, Whistler, BC, Canada, 2010.

[100] S. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in Neural Information Processing Systems,* vol. 19, 2007, pp. 153-160.

[101] G. Hinton, "Learning multiple layers of representation," *Trends in Cognitive Sciences,* vol. 11, 2007, pp. 428-434.

[102] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE,* vol. 86, 1998, pp. 2278-2324.

[103] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature,* vol. 518, 2015, pp. 529-533.

[104] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature,* vol. 529, 2016, pp. 484-489.

[105] L. van Hateren and A. van der Schaaf, "Independent Component Filters of Natural Images Compared with Simple Cells in Primary Visual Cortex," *Proceedings: Biological Sciences,* vol. 265, 1998, pp. 359-366.

[106] P. Hoyer and A. Hyvärinen, "Independent Component Analysis Applied to Feature Extraction from Colour and Stereo Images," *Computation in Neural Systems,* vol. 11, 2000, pp. 191-210.

[107] Y. LeCun and Y. Bengio, "Convolutional Networks for Images, Speech, and Time Series," in *The Handbook of Brain Theory and Neural Networks*, ed: MIT Press, 1995.

[108] K. Fukushima, "Neocognitron for handwritten digit recognition," *Neurocomputing,* vol. 51, 2003, pp. 161-180.

[109] D. Hubel and T. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol,* vol. 106, 1962, pp. 106-154.

[110] D. Hubel and T. Wiesel, "Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat," *J. Neurophysiol.,* vol. 28, 1965, pp. 229-289.

[111] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun, "Learning invariant features through topographic filter maps," *Proceedings of* IEEE Conference on Computer Vision and Pattern Recognition 2009 (CVPR 2009), 2009.

[112] H. Goh, L. Kusmierz, J.-H. Lim, N. Thome, and M. Cord, "Learning Invariant Color Features with Sparse Topographic Restricted Boltzmann Machines," *18th IEEE Conference on Image Processing (ICIP)*, 2011, pp. 1241-1244.

[113] A. Hyvärinen, J. Hurri, and P. Hoyer, *Natural Image Statistics*: Springer, 2009.

[114] Q. Le, W. Zou, S. Yeung, and A. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," *CVPR 2011*, 2011.

[115] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, 2009, pp. 248-255.

[116] J. Weston, S. Bengio, and N. Usunier, "Wsabie: Scaling up to large vocabulary image annotation," *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, Barcelona, Spain, 2011, pp. 2764-2770.

[117] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller. (2007). *Labeled faces in the wild: A database for studying face recognition in unconstrained environments*. Available: http://vis-www.cs.umass.edu/lfw/

[118] W. Zhang, J. Sun, and X. Tang, "Cat head detection - how to effectively exploit shape and texture features," *Proceedings of the 10th European Conference on Computer Vision (ECCV)*, Marseille, France, 2008, pp. 802-816.

[119] C. Keller, M. Enzweiler, and D. M. Gavrila, "A new benchmark for stereo-based pedestrian detection," *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2009, pp. 691-696.

[120] P. Földiák, "Learning Invariance from Transformation Sequences," *Neural Computation,* vol. 3, 1991, pp. 194-200.

[121] W. Einhäuser, C. Kayser, P. König, and K. P. Körding, "Learning the Invariance Properties of Complex Cells from their Responses to Natural Stimuli," *European Journal of Neuroscience,* vol. 15, 2002, pp. 475-486.

[122] W. Zou, S. Zhu, A. Ng, and K. Yu, "Deep Learning of Invariant Features via Simulated Fixations in Video," *Advances in Neural Information Processing Systems (NIPS)*, Lake Tahoe, Nevada, USA, 2012, pp. 3203-3211.

[123] G. Hinton and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science,* vol. 313, 2006, pp. 504-507.

[124] L. van Hateren and J. Ruderman, "Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex," *Proceedings of the Royal Society B: Biological Sciences,* vol. 265, 1998, pp. 2315-2320.

[125] H. Wersing and E. Korner, "Learning Optimized Features for Hierarchical Models of Invariant Object Recognition," *Neural Computation,* vol. 15, 2003, pp. 1559-1588.

[126] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *CVPR Workshop on Generative-Model Based Vision*, 2004.

[127] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," *International Conference on Machine Learning*, 2010, pp. 111-118.

[128] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," *Master's Thesis*, Dept. of Comp. Sci., University of Toronto, 2009.

[129] F. J. Huang and Y. LeCun. (2004). *The NORB Dataset*. Available: http://www.cs.nyu.edu/~ylclab/data/norb-v1.0/

[130] Y. Tang, "Deep Learning using Linear Support Vector Machines," *Workshop on Challenges in Representation Learning, ICML*, 2013.

[131] A. Elgammal, *Background Subtraction: Theory and Practice*: Morgan & Claypool Publishers, 2014.

[132] A. Barjatya, "Block Matching Algorithms for Motion Estimation," Utah State University, *Technical Report*, 2004.

[133] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '98)*, 1998, pp. 232-237.

[134] P. Torr and A. Zisserman, "Feature Based Methods for Structure and Motion Estimation," *Proceedings of the International Workshop on Vision Algorithms*, Corfu, Greece, 2000, pp. 278-294.

[135] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering,* vol. 82, 1960, pp. 35-45.

[136] C. M. Kuo, "Motion estimation for video compression using Kalman filtering," *IEEE Transactions on Broadcasting,* vol. 42, 1996, pp. 110-116.

[137] S. Erturk, "Digital image stabilization with sub-image phase correlation based global motion estimation," *IEEE Transactions on Consumer Electronics,* vol. 49, 2003, pp. 1320-1325.

[138] L. Matthies, T. Kanade, and R. Szeliski, "Kalman filter-based algorithms for estimating depth from image sequences," *International Journal of Computer Vision,* vol. 3, 1989, pp. 209-238.

[139] V. Mahadevan and N. Vasconcelos, "Biologically Inspired Object Tracking Using Center-Surround Saliency Mechanisms," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 35, 2013, pp. 541-554.

[140] N. Wang and D. Yeung, "Learning a Deep Compact Image Representation for Visual Tracking," *Advances in Neural Information Processing Systems*, 2013, pp. 809-817.

[141] M. Norouzi, M. Ranjbar, and G. Mori, "Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning," *IEEE Conference on Computer Vision and Pattern Recognition 2009 (CVPR 2009)*, 2009, pp. 2735-2742.

[142] M. Marszalek, I. Laptev, and C. Schmid, "Actions in Context," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 2929-2936.

[143] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?," *Journal of Machine Learning Research,* vol. 11, 2010, pp. 625-660.

[144] B. A. Wandall, "Visual field maps in human cortex," *Neuron,* vol. 56, 2007, pp. 366-383.

[145] J. H. Kaas, "The evolution of complex sensory and motor systems of the human brain," *Brain Research Bulletin,* vol. 75, 2008, pp. 384-390.

[146] L. Isik, J. Z. Leibo, and T. Poggio, "Learning and Disrupting Invariance in Visual Recognition with a Temporal Association Rule," *Frontiers in Computational Neuroscience,* vol. 6, 2012.

[147] E. T. Rolls, "Neurophysiological mechanisms underlying face processing within and beyond the temporal cortical visual areas," *Philosophical Transactions of the Royal Society,* vol. 335, 1992, pp. 11-21.

[148] P. Földiák, "Models of sensory coding," University of Cambridge, Department of Engineering, 1992.

[149] E. T. Rolls and M. J. Tovee, "Processing speed in the cerebral cortex and the neurophysiology of visual masking," *Proceedings of the Royal Society B: Biological Sciences,* vol. 257, 1994, pp. 9-15.

[150] I. J. Goodfellow, D. Warde-Farley, P. Lamblin, V. Dumoulin, M. Mirza, R. Pascanu, J. Bergstra, F. Bastien, and Y. Bengio, "Pylearn2: a machine learning research library," *arXiv preprint arXiv:1308.4214*, 2013.

[151] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring, *GNU Octave version 4.0.0 manual: a high-level interactive language for numerical computations*, 2015.

[152] H. Larochelle and S. Bengio, "Classification using discriminative restricted Boltzmann machines," *International Conference on Machine Learning*, 2008, pp. 536-543.

[153] I. Goodfellow, Q. Le, A. Saxe, H. Lee, and A. Ng, "Measuring Invariances in Deep Networks," *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 646-654.

[154] A. Yogeswaran and P. Payeur, "Improving Visual Feature Representations by Biasing Restricted Boltzmann Machines with Gaussian Filters," in *Advances in Visual Computing: 12th International Symposium, ISVC 2016, Las Vegas, NV, USA, December 12-14, 2016, Proceedings, Part I*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, F. Porikli, S. Skaff*, et al.*, Eds., ed Cham: Springer International Publishing, 2016, pp. 825-835.

[155] A. Yogeswaran and P. Payeur, "Biasing restricted Boltzmann machines using Gaussian filters to learn invariant visual features," *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*, 2016, pp. 1-8.

[156] M. S. Fine and B. S. Minnery, "Visual salience affects performance in a working memory task," *Journal of Neuroscience,* vol. 29, 2009, pp. 8016-8021.

[157] M. Joëls, Z. Pu, O. Wiegert, M. S. Oitzl, and H. J. Krugers, "Learning under stress: how does it work?," *Trends in cognitive sciences,* vol. 10, 2006, pp. 152-158.

[158] L. Cahill, B. Prins, M. Weber, and J. L. McGaugh, "ß-Adrenergic activation and memory for emotional events," *Nature,* vol. 371, 1994, pp. 702-04.

[159] A. Rahimi, L.-P. Morency, and T. Darrell, "Reducing drift in parametric motion tracking," *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, 2001, pp. 315-322.

[160] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 1999, pp. 246-252.

[161] Z. Xianyi, W. Qian, and Z. Chothia, "OpenBLAS," *URL: http://xianyi. github. io/OpenBLAS*, 2014.

[162] G. Bradski, "The OpenCV Library (2000)," *Dr. Dobbs J. Softw. Tools Prof. Program,* vol. 1, 2000, pp. 1-6.

[163] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: a structure for efficient numerical computation," *Computing in Science & Engineering,* vol. 13, 2011, pp. 22-30.

[164] R. Brunelli, *Template matching techniques in computer vision: theory and practice*: John Wiley & Sons, 2009.

[165] A. Frost and M. Niemeier, "Suppression and reversal of motion perception around the time of the saccade," *Frontiers in Systems Neuroscience,* vol. 9, 2015.

[166] E. Castet, S. Jeanjean, and G. S. Masson, "Motion perception of saccade-induced retinal translation," *Proceedings of the National Academy of Sciences,* vol. 99, 2002, pp. 15159-15163.

[167] J. B. Benson and M. M. Haith, *Language, memory, and cognition in infancy and early childhood*: Academic Press, 2010.

[168] A. Yogeswaran and P. Payeur, "Leveraging Saccades to Learn Smooth Pursuit: A Self-Organizing Motion Tracking Model Using Restricted Boltzmann Machines," *Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, San Francisco, California, USA, 2017, pp. 4313-4319.

[169] A. Yogeswaran and P. Payeur, "Self-Organizing Smooth Pursuit Model using Retinal Constancy," *Neural Computation*, 2017 (submitted).

[170] I. Sutskever, G. E. Hinton, and G. W. Taylor, "The recurrent temporal restricted boltzmann machine," *Advances in Neural Information Processing Systems*, 2009, pp. 1601-1608.