# ON THE STUDY OF DEEP LEARNING ACTIVE VISION SYSTEMS

PIOTR ALEKSANDER OZIMEK

SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
*Doctor of Philosophy*

SCHOOL OF COMPUTING SCIENCE

COLLEGE OF SCIENCE AND ENGINEERING
UNIVERSITY OF GLASGOW

JULY 2022

## Abstract

This thesis presents a series of investigations into various active vision algorithms. An experimental method for evaluating active vision memory is proposed and used to demonstrate the benefits of a novel memory variant called the WW-LSTM network. A method for training active vision attention using classification gradients is proposed and a proof of concept of an attentional spotlight algorithm is demonstrated to convert spatially arranged gradients into coordinate space. The thesis makes a number of empirically supported recommendations as to the structure of future active vision architectures.

Chapter 1 discusses the motivation behind pursuing active vision and lists the objectives set out in this thesis. The chapter contains the thesis statement, a brief overview of the relevant background and a list of the main contributions of this thesis to the literature.

Chapter 2 describes an investigation into the utility of the software retina algorithm within the active vision paradigm. It discusses the initial research approach and motivations behind studying the retina, as well as the results that prompted a shift in the focus of this thesis away from the retina and onto active vision. The retina was found to slow down training to an infeasible pace, and in a latter experiment it was found to perform worse than a simple image cropping algorithm on an image classification task.

Chapter 3 contains a comprehensive and empirically supported literature review highlighting a number of issues and knowledge gaps present within the relevant active vision literature. The review found the literature to be incoherent due to inconsistent terminology and due to the pursuit of disjointed approaches that do not reinforce each other. The literature was also found to contain a large number of pressing knowledge gaps, some of which were demonstrated experimentally. The literature review is accompanied by the proposal of an investigative framework devised to address the identified problems in the literature by structuring future active vision research.

Chapter 4 investigated the means by which an active vision systems can collate the information they obtain across multiple observations. This aspect of active vision is referred to as memory. An experimental method for evaluating active vision memory in an interpretable

manner is devised and applied to the study of a novel approach to recurrent memory called the WW-LSTM. The WW-LSTM is a parameter-efficient variant of the LSTM network that outperformed all other recurrent memory variants that were evaluated on an image classification task. Additionally, spatial concatenation in the input space was found to outperform all recurrent memory variants, calling into question a commonly employed approach in the active vision literature.

Chapter 5 contains an investigation into active vision attention, which is the means by which the system decides where to look. Investigations contained therein demonstrate the benefits of employing a curriculum for training attention that modifies sensor parameters, and present an empirically backed argument in favour of implementing attention in a separate processing stream from classification. The chapter closes with a proposal of a novel method for leveraging classification gradients in training attention; the method is called predictive attention, and a first step in its pursuit is taken with a proof of concept demonstration of the hardcoded attention spotlight algorithm. The spotlight is demonstrated to facilitate the localisation of a hotspot in a modelled feature map via an optimisation process.

Chapter 6 concludes this thesis by re-stating its objectives and summarizing its key contributions. It closes with a discussion of recommended future work that can further advance our understanding of active vision in deep learning.

## Acknowledgements

I thank my first Ph.D. supervisor, Paul Siebert, for getting me started on this journey, showing me that rules can be broken, and raising my ambitions so high.

My second Ph.D. supervisor, Gerardo Aragon Camarasa, went out of his way to take up my supervision at a critical moment during this Ph.D. project. Gerardo's patience, attitude and guidance kept my worst tendencies in check; without him, I would not be as proud of my work as I am today. I am extremely thankful to him for everything.

I am grateful to John Williamson and Ke Yuan for critically reviewing my work throughout the years. Their negative feedback pushed me to reformulate my goals and do better.

I need to thank my family, especially my parents, for supporting me, for staying in touch, and for never missing a chance to ask me how much longer I will work on this thesis or complain about how I am taking too long writing.

Special thanks to my friends Fu, Iulia, Marcin, Zuzanna and Jakub for keeping me sane, understanding me, and keeping my hope alive. Also, thanks to my online friends BobbaBear and Slyhagr0 for helping me stave off boredom.

Finally, I must thank Veronika for being the reason why I decided to do a Ph.D. in the first place and for giving me the wings that still carry me today.

To Veronika and the mumies.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

***Abstract:*** *The goal of this thesis is to advance the current understanding of active vision in deep learning by developing and investigating deep learning active vision architectures using foveated virtual sensors. The thesis aims to accomplish this by proposing a research framework for investigating active vision and by investigating a number of different approaches to active vision.*

## 1.1 Thesis Statement

An overwhelming majority of computer vision systems in use today process images and videos in a passive fashion, often by exhaustively scanning them in entirety using the convolution operator. In contrast, animals observe their environments by actively selecting areas to focus on. *How can we enable deep learning vision systems to 'see' by actively exploring the scene?* This thesis first grounds the above question with a research framework that elucidates the various aspects of active vision; it then pursues the answer by investigating a series of different algorithms focusing on an active vision system's sensor, memory and attention. The purpose of this thesis is to advance the current understanding of active vision in deep learning by developing and investigating a deep learning active vision architecture.

## 1.2 Motivation

A vision system is said to be active if it modulates the captured signal by controlling the geometrical parameters of its visual sensor in order to improve its performance [4]. In contrast, passive vision systems sample from the scene in a uniform fashion and do not intentionally alter the geometry of their visual sensor to improve their performance. The vast majority of approaches to vision in deep learning are passive, whereas in nature there are no known examples of passive vision. Therefore, the first motivation for the pursuit of active vision is biomimetics, which is the idea of mimicking biological processes when solving technological problems. The fact that many aspects of biological vision systems are still superior to computer vision is sufficient motivation for biomimicry.

While a significant proportion of computer vision systems in use today processes disembodied images that were captured by passive sensors with no capacity or need to interact with their environments, the scope of computer vision goes well beyond that. Computer vision is also a necessary tool in the development of autonomous robotic systems which have the potential to revolutionize multiple aspects of our society. There are reasons to believe that rejecting passive vision in favour of active vision is preferable, if not unavoidable, in the pursuit of autonomy in robotics and AI. Various robotic tasks, such as searching or navigating a complex, real world environment, can involve processing vast amounts of visual information. Robotic systems are frequently subject to various constraints, whether relating to the amount of available computational power or the time available to complete their task, that necessitate reducing the amount of visual information they process. For this reason they can rarely afford to maintain and process an exhaustive model of their environment or to continuously monitor a high-resolution 360 degree view of their surroundings, as would be warranted by a passive vision system operating in an unconstrained environment. By enabling selective and intelligent sampling from the environment, active vision has the potential to greatly reduce the amount of visual information that an autonomous agent has to process in order to complete its tasks.

The author of this thesis believes that a significant amount of research efforts in deep learning are misdirected to developing solutions that excel on highly controlled research datasets but struggle in less constrained, real-world settings. A field-wide switch to experimenting in real-world environments is not a feasible solution as it would drastically reduce the volume of vision research performed. Instead, imposing additional constraints on either the vision system's function or on the dataset used can bring the development setting of vision algorithms closer to the real-world setting. Using an active vision system with a foveated sensor and a limited field of view is one such set of constraints, which is why active vision has the potential to lead to the development of vision systems that are more robust and more suitable for real-world environments.

Most deep learning datasets include a strong human bias as they consist of images taken by people concerned with the photographs' aesthetic appeal. Such photographs tend to follow the ”rule of thirds” [5], position clutter in the background and have the object of interest viewed from a favourable angle. Datasets designed to avoid this bias [6] [7] [8] are scarce, costly to collect and receive very little research attention. Active vision grants the system control over how an object of interest is viewed, which is analogous to letting the system imbue an input image with its own biases. The ability to add its own algorithmic bias could hypothetically improve the system's invariance to human biases that are present in its training data.

In summary, the pursuit of active vision is motivated by the performance of biological vision systems, by the potential of active vision to reduce the volume of information processed by autonomous agents, by its potential to lead to more robust vision systems without experimenting in real-world environments, and by enabling vision systems to overcome human biases present in the training data. These motivations are theoretical and verifying them empirically is an appealing research objective; however, active vision in deep learning is too immature and incoherent as a research field for such an investigation to be feasible within the scope of this thesis. Instead, the following objectives were established to guide the research conducted in this Ph.D.:

- To investigate the suitability of a retina-like biomimetic sensor for active vision systems.
- To establish and follow a research practice that can address the knowledge gap present in the active vision literature.
- To produce foundational knowledge that can guide future active vision research.
- To investigate how does an active vision system collate the information that it obtains across multiple observations.
- To investigate different approaches in enabling an active vision system to decide where to look.

## 1.3 Background

### 1.3.1 The Software Retina

Although advancing active vision is the overarching objective of this thesis, the main focus at the onset of this Ph.D. was the software retina. The software retina is a loosely biomimetic algorithm for subsampling images in a space-variant manner. It consists of a sampling structure that aims to mimic the foveated distribution of retinal ganglion cells in the human eye, with the input pixels corresponding to photoreceptors that fall within their receptive fields [3]. Section 2.2.2 provides a more detailed description of the software retina algorithm.

The initial objective of this thesis was investigating the software retina by integrating it with deep learning vision systems. One such integration, dubbed the cortical mapping, has been described in the literature prior to the research carried out in this thesis [1]. The cortical

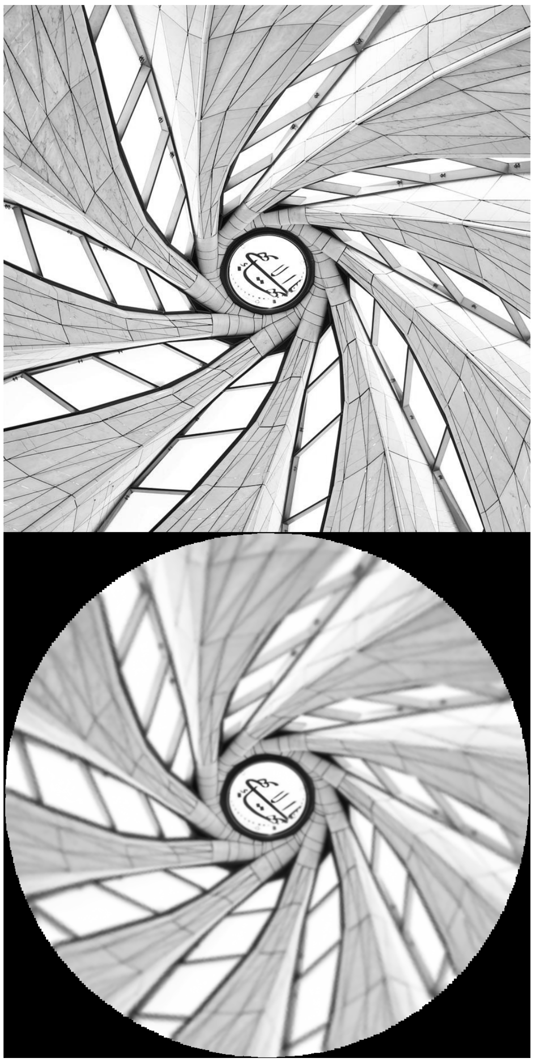


Figure 1.1: Top: an input image. Bottom: a backprojection image that visualises the data captured by the software retina. Taken from [1]. An explanation of the role of the image can be found in Section 2.2.2.

mapping algorithm was inspired by the spatial transformation undergone by visual signals as they travel from the retina to the primary visual cortex [2]. The goal of the mapping is to address the fact that the software retina's output format is not appropriate for processing with the convolution operator; it does that by producing a spatially efficient and conformal projection of the data captured by the retina. More details about the cortical mapping algorithm can be found in Section 2.2.3.

Investigating the software retina and the cortical mapping was motivated by their capacity to produce a compact representation of a wide field of view, which can help reduce the computational costs incurred by deep learning systems processing their data. Additionally, the software retina presents itself as a potential platform for emulating numerous neural computations that take place in animal retinas. More hypothetical motivations were involved in driving this line of research; however, as the research conducted in this thesis has progressed

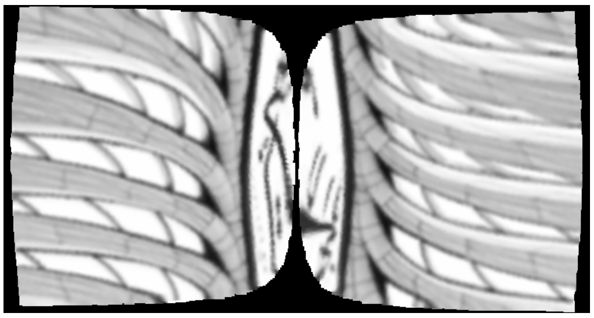


Figure 1.2: A cortical image produced from the input image shown at the top of Figure 1.1. Not to scale. Taken from [1]. An explanation of the significance of the cortical image can be found in Section 2.2.3.

it was decided that shifting the focus away from the software retina and onto active vision was necessary. Chapter 2 covers the rationale behind this decision.

### 1.3.2 Active Vision in Deep Learning

Within computer vision, research interest in active vision has been initiated by Aloimonos et. al. [4] and Ballard [9]. Aloimonos et. al. have motivated active vision by showing that various low-level visual problems, such as structure from motion, shape from shading, shape from texture and shape from contours are more tractable with an active observer capable of perceiving the object of interest from different viewpoints [4]. Ballard has argued for the benefits of active vision with foveated gaze control, one of which is that various vision tasks involving motion are simplified when using an exocentric coordinate frame, i.e. when the observer is capable of locking their gaze onto a moving object and tracking it [9].

In spite of active vision being a relatively active research domain in the past, interest in it has waned as interest in deep learning started growing circa 2012. As a result, literature focusing on active vision in deep learning is difficult to come by. The literature also suffers from numerous other issues that are discussed in more detail in Section 3.2.7. One such issue is the scarcity of coherent and continuous research threads where successive publications meaningfully advance prior work. Research focusing on the Recurrent Attention Model (RAM) architecture represents an exception to this trend, as it is the most mature and actively researched active vision deep learning architecture at the time of writing this thesis. The architecture was initially proposed by Mnih et. al. [10]; however, various variants have been developed and investigated since [11] [12] [13] [14] [15] [16]. All RAM variants follow the same broad design and consists of a virtual sensor that crops out foveated image patches from the scene and passes them to a feature extractor; these features are provided to a recurrent network whose hidden state is used in image classification and in driving the virtual sensor to further explore the scene. A more detailed description of the RAM architecture can be found in Section 3.2.2. As it is the most actively developed architecture in the literature, it has been the architecture of choice for a majority of the experiments conducted in this thesis.

## 1.4 Active Vision Research Framework

The literature review in Section 3.2 together with the experiments in Section 3.3 identify numerous shortcomings within the active vision in deep learning literature. Arguably, all of these shortcomings can be reduced into two underlying problems: an incoherence across the field and a large number of research and knowledge gaps pertaining to key issues. To address these issues this thesis proposes an investigative framework for structuring active vision research. The framework aims to coin standard terminology for use in active vision research, to make future literature more coherent and easier to browse. It also strives to promote a perspective on active vision that highlights numerous pressing knowledge gap that the literature has previously failed to address.

The framework breaks down passive vision systems into two components: the feature extractor and the mechanism for using the extracted features to complete the system's visual task. The framework then defines an active vision system as complementing these two components with three additional architectural requirements: a sampling structure that defines the active agent's local sampling strategy, an attention mechanism which implements the agent's global sampling strategy, and a memory mechanism that implements the agent's strategy for aggregating visual data collected across multiple observations. The research conducted in Chapters 4 and 5 has been structured in accordance with the prescriptions produced by the framework. A full description of the framework and a discussion of various interrelations between the three aspects of active vision can be found in Section 3.4.

## 1.5 Contributions

The key contributions of this thesis towards advancing our understanding of active vision in deep learning are:

- An investigation into the utility of the software retina algorithm within the active vision paradigm that produced negative results, demotivating its pursuit and demonstrating the necessity of researching active vision in a more structured manner and as a matter of precedence.
- A comprehensive and empirically supported commentary on the issues with the relevant active vision literature, accompanied by an investigative framework devised to address the identified problems by guiding future active vision research.

- An experimental method for evaluating active vision memory that enables producing interpretable results without having to devise a solution to the problem of attention.
- A novel variant of the LSTM architecture, called the WW-LSTM, that functions as a active vision recurrent memory mechanism that collates information obtained across multiple observations.
- A number of empirically supported recommendations as to the structure of future active vision architectures.
- An investigation into the hardcoded attention spotlight algorithm, that functions as a stepping stone towards a novel method for training active vision attention with classification gradients.

All research code used to conduct the experiments described in this thesis can be found at `https://github.com/Pozimek/CUB-RAM`. Research presented in this thesis has also been described as a part of the following publication:

- Ozimek, P., Hristozova, N., Balog, L., & Siebert, J. P. (2019). A space-variant visual pathway model for data efficient deep learning. Frontiers in cellular neuroscience, 13, 36.

## 1.6 Thesis Outline

The remainder of this thesis is organised in the following way:

- **Chapter 2** introduces the software retina, reviews the retina-related literature and describes the exploratory research that has been conducted with the retina as its focus. It also explains the motivation behind suspending the pursuit of the software retina.
- **Chapter 3** reviews the literature relevant to active vision in deep learning, conducts experiments that demonstrate the knowledge gap and proposes a research framework to address the issues with the literature.
- **Chapter 4** proposes an experimental approach to investigating active vision memory and employs it to evaluate several recurrent memory variants.
- **Chapter 5** investigates active vision attention by exploring a pair of research questions and by proposing a novel method for training attention.
- **Chapter 6** closes this thesis with conclusions and a discussion of future work.

# Chapter 2

# Exploratory Work Using the Software Retina

***Abstract:*** *The main investigations in this thesis focus on the pursuit and study of active vision; however, the starting goals of this Ph.D. were different. This chapter opens with a description of the initial research approach and motivations that focus on the software retina. It then describes the function of the software retina and reviews the literature that is relevant to it. Finally, the chapter describes the exploratory work done in accordance with the initial research approach, the problems encountered and how they prompted a revision of the project scope that lead to the research framework described in Section 3.4. The exploratory work consisted of developing a colour opponency model described in Section 2.3.2 and the pilot study performed on the EPIC Kitchens dataset described in Section 2.4.*

## 2.1 Motivation and Objectives

The primary goal at the beginning of the project was the development and investigation of the software retina for deep learning vision systems. The software retina is a functional model of the human retina's foveated architecture that utilizes Gaussian receptive fields to implement a space variant sampling strategy [3]. A complete technical description of the software retina can be found in Section 2.2.2. In the context of deep learning it can be understood as a virtual sensor that is driven around the input image as though it is exploring a scene that spans beyond its field of view. It acts as a pre-processing step applied to all image data passed to the neural network and it stands in stark contrast to the passive vision paradigm, which is how conventional deep learning vision systems process image data by uniformly processing the entire scene.

One of the main motivations for studying the software retina was investigating whether biomimetics, which is the practice of mimicking biological systems for solving human problems, can advance the state of computer vision literature. Deep learning algorithms are a suitable match for the software retina owing to them being the state-of-the-art solution for a wide range of vision problems that also is a biomimetic approach loosely inspired by how the brain works. The retina was hypothesized to reduce the computational costs and data requirements associated with deep learning. State-of-the-art architectures consist of 100s of millions of parameters and can take over a hundred GFLOPs to complete a forward pass [17]. This may not be a significant challenge for vision systems deployed in data centres or the cloud; however, it does pose a problem for mobile and robotic architectures that are subject to much more austere hardware constraints.

One motivating hypothesis was that *utilizing foveated sampling, as implemented in the software retina, has the potential to alleviate the computational costs of deep learning. Human vision demonstrates that perceiving the complete object of interest in high resolution is not required in order to recognize it; it is sufficient for us to fixate our foveae on key locations of the object and only dedicate minimal computational resources associated with the peripheries of our vision to the less salient parts of the object. This implies that uniformly sampling the entire scene in full resolution, as is commonly done in most deep learning vision systems today, is a highly redundant approach in need of optimization.*

The primary goal of this chapter is to investigate the software retina's integration with deep learning vision systems. In accordance with the pursuit of biomimetics, another aim was to utilize the software retina as a platform for replicating other functions of biological retinas. The plan for early research was to expand the functional capabilities of the retina so as to enable more hypotheses to be tested later, once the retina was already deployed in an deep learning environment suitable for rigorous investigation. The remainder of this chapter explains why and how this approach has been revised.

## 2.2 Background

This section provides the context required for the exploratory work described in the remainder of the chapter. In order to ground the biomimetic theme of the software retina the first part of this section covers the relevant aspects of mammalian vision systems. At the onset of this project the software retina was at a very early stage of research and development as only two prior projects advanced it: the work of Balasuriya [3] which conceived the software retina and the work of Ozimek and Siebert [1] which investigated a rudimentary integration of the retina with convolutional neural networks. The next part of this section will focus on the first of these works and the software retina.

An integration between the software retina and deep learning is necessary and non-trivial, as the output data of the software retina is in a format that is not compatible with the convolution operator that forms the foundation of most deep learning vision systems. At the time of writing the vision transformer architecture provides an alternative to the convolution operator that is competitive with the state-of-the-art on a number of benchmarks [18]. Vision transformers could therefore offer an alternative pathway for integrating the retina with deep learning vision systems; however, they only became available quite late during this project and thus could not be made a part of these investigations. The final part of this section describes the retino-cortical transform, which implements the first integration between the software retina and convolutional neural networks.

### 2.2.1 Biological Vision

In order to be perceived, any light entering the eyeball has to stimulate a layer of photoreceptor cells located at the rear of the eye. In humans these cells are arranged in a foveated pattern, meaning that their distribution is the densest in the centre of the retina - dubbed the fovea - and is progressively sparser towards the peripheries [19]. Curiously, in humans photoreceptors are facing inwards towards our heads, so that any light that stimulates them first passes through the other neuronal layers in the retina. There are two types of photoreceptor cells: rods and cones. Rods are very light-sensitive and thus operate well in dim light but are colour-blind. Cones require relatively intense light to activate and each cone is selectively sensitive to either short, medium or long wavelength light [20].

As light stimulates photoreceptor cells they produce signals that are pre-processed by up to 4 different neuron types before leaving the retina along the optic nerve. These 4 neuron types are the horizontal cells, bipolar cells, amacrine cells and retinal ganglion cells, each having numerous subtypes, with the axons of the ganglion cells forming the optic nerve connecting the eye with the brain. There are at least 18 different types of retinal ganglion cells present in the primate retina with each one of them being functionally distinct and transmitting a different type of signal. The functional characteristics of these different ganglion cells stem from the horizontal, amacrine and bipolar cells that feed into them [21] [22]. These neurons perform a wide range of functions, including but not limited to brightness constancy, detecting approaching motion, motion extrapolation, discriminating the direction of texture and object motion and anticipating signal periodicity [23].

The signals captured by different retinal ganglion cell species travel along their axons into the primary visual cortex (V1). Along the way these signals undergo a spatial transformation; the optic nerves from each eye cross in the optic chiasm where they are split into two parts, each corresponding to one half of the animal's visual field. The rearranged optic nerves then travel through the lateral geniculate nucleus and arrive inside V1 in a log-polar-like retinotopic mapping (Figure 2.1). Owing to evolutionary pressures on their vision systems,

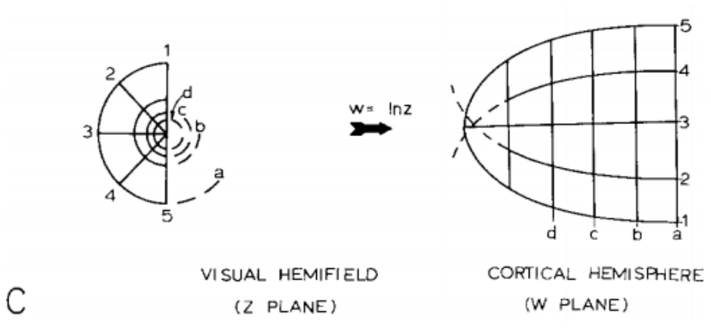


Figure 2.1: A generalized representation of the global retinotopic mapping inside V1, taken from [2]

different animals have different distributions of photoreceptors and retinal ganglion cells that in turn give rise to different retinotopic mappings in their visual cortices [2].

Colour opponency is a functional characteristic of biological vision that plays a significant role in V1 but originates from within the retinal circuitry [24] [25]. Within much of the visual system colour information is encoded in an opponent fashion where the lack of a colour-sensitive neuron's activation can code for another colour, as opposed to an absolute encoding where the lack of a response would always mean a complete lack of stimulus. In other words, opponent cells are activated by signals coming from cones sensitive to one colour and suppressed by signals coming from cones sensitive to another. There are two known types of colour opponent cells: single-opponent cells and double-opponent cells [24]. Single-opponent cells simply exhibit colour opponent behaviour as previously described between the colour pairs of red-green, blue-yellow and black-white. Double-opponent cells have a response profile that is inverted between their receptive field's centre and surround; the colour that activates the cell in the centre of its receptive field suppresses it in its peripheries and vice versa. Section 2.3.2 describes how the function of these cells was mimicked within the software retina.

### 2.2.2 The Software Retina

The software retina is a structure and an associated algorithm for subsampling images in a space-variant manner. Generating its structure only has to be done once and does not factor into the algorithm's runtime. The sampling structure aims to mimic the foveated distribution of retinal ganglion cells in the human eye, with the input pixels corresponding to photoreceptors that fall within their receptive fields. It consists of a circular arrangement of point locations (Figure 2.2) and their associated subsampling kernels (Figure 2.3). Each point location defines the subpixel centre of a Gaussian kernel used to subsample from the input image, whereas the local density of neighbouring points defines the kernel's parameterisation. Originally the arrangement of point locations was generated using a self-similar neural

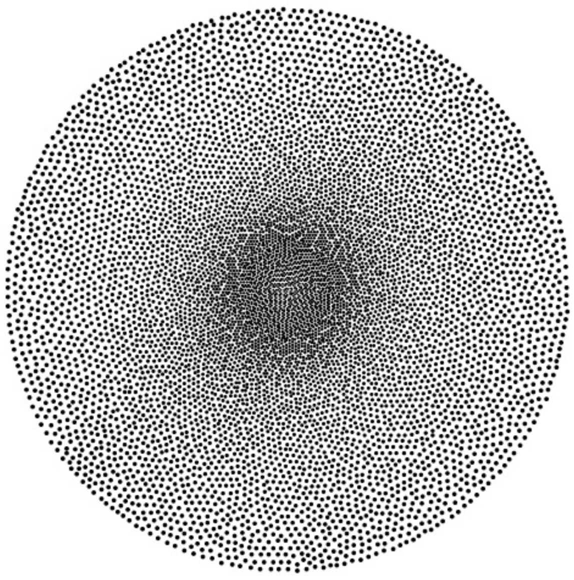


Figure 2.2: Point locations defining the receptive field centres of the software retina.

network to ensure that no local discontinuities or distortions are present in the structure [26] [3]; however, in theory it can also be produced using other algorithms such as logarithmic spiral equations.

To subsample from the image the structure is centred on a user-defined fixation location, the Gaussian kernels are multiplied with the pixels that they overlay and their responses are summed together and returned in the image vector data structure [3]. The image vector is an NxC tensor, where N is the number of receptive fields in the software retina and C is the number of colour channels in the input image. Although each image vector value is associated with a Gaussian kernel that has a known location, such explicit location encoding is not compatible with the convolution operator and thus the integration of the software retina with convolutional neural networks is not trivial. The visual data stored in the image vector can be visualized by passing it back through the sampling structure onto an image grid to produce a backprojected image (Figure 2.4). This visualisation can be used to help diagnose any issues present in the generated sampling structure of the software retina. The backprojected image can also be used as input to convolutional neural networks; however, it would be difficult to motivate such a design choice as the computational overhead of the software retina algorithm is quite significant while the hypothesized benefits of using the backprojected image are difficult to devise.

The software retina has a number of hypothesized benefits when understood as a sampling algorithm to be used in tandem with a transforming algorithm for presenting the captured data to a convolutional neural network, such as the one described in Section 2.2.3. The retina and any transforming algorithm used would impose their own computational overheads in a deep learning system; however, the image vector is a compact representation of the input data that

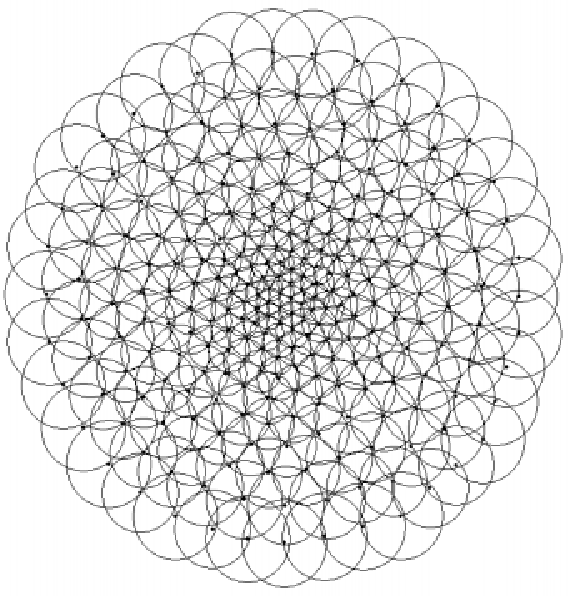


Figure 2.3: A visualisation of a small retina's Gaussian receptive fields, taken from [3]

could enable a significant reduction in the computational cost of a neural network's forward pass if used with a transforming algorithm that preserves the compression. The compression ratio of the image vector improves with the retina's size, thus a large retina would stand a chance of more than offsetting its own overheads by reducing the computational costs of any neural network that utilizes its output.

The simplest alternative to the software retina that sub-samples from an input image in a foveated manner consists of an algorithm that extracts overlapping patches at varying resolutions [10]. The software retina is more computationally expensive in comparison; however, its structure enables a smooth representation of visual features across multiple sampling frequencies; it is effectively a continuous wide range bandpass filter while the approach utilizing overlapping patches is a discrete, narrow and non-overlapping bandpass filter that is likely to corrupt features present at the border of the foveal patch. Another benefit of the software retina is that the receptive field structure enables using task-specific sampling distributions that are different to the human foveation pattern as well as a spatial transformation of the captured image such as the one described in 2.2.3 that may have its own advantages.

The receptive field structure also makes possible the selective mimicking of computations that take place in the mammalian retina and give rise to different retinal ganglion cell types. One example function that could be reproduced is that of the Y-type ganglion cells; they are involved in detecting texture motion, i.e. when a regular pattern is translated across the retina's field-of-view, in a direction-insensitive but a velocity-sensitive manner [27]. These cells are thought to play a role in the segmentation of moving objects [23]. A lot of the neural circuitry in the retina appears to compute temporal features; the starburst amacrine cells selectively inhibit bipolar cells depending on their temporal activations [28], while

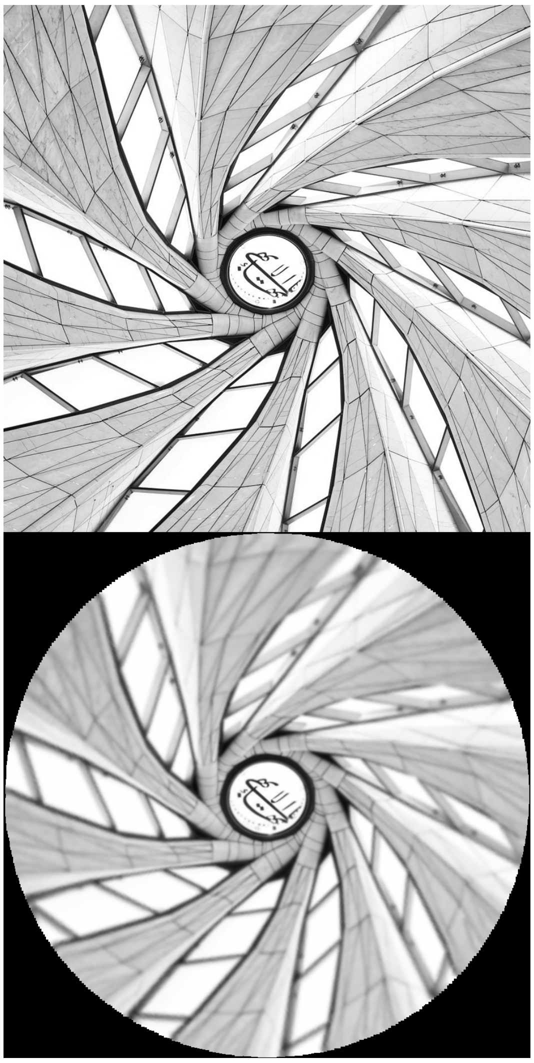


Figure 2.4: Top: an input image. Bottom: retinal backprojection image. Taken from [1].

the Object Motion Sensitive retinal ganglion cells detect differential motion, i.e. motion that differs from the surrounding motion [29]. In theory a neural network could implicitly learn to mimic these computations; however, neural network training is a highly stochastic process and there is no guarantee for this to happen, thus hard-coding these early stage vision computations into the vision system could be of value and would be a justified topic of investigation in line with the project's motivation. In practice these research ideas were left as future work for reasons described in Section 2.3.2.

### 2.2.3 The Cortical Mapping

Realizing and evaluating the hypothesized benefits of the software retina in a deep learning system is contingent upon an effective presentation of the data captured in the image vector. At the onset of this project the cortical mapping algorithm introduced in [1] was the only solution available for this task.

This algorithm was inspired by the spatial transformation undergone by visual signals as they travel from the retina to V1 [2] and its goal is to produce a structure capable of spatially efficient and conformal projection of the data captured by the retina. Just like in the optic nerve, the cortical mapping algorithm splits the receptive field structure of the software retina into two halves along the vertical meridian. The two sets of point locations are then spatially transformed according to formulas 2.1 and 2.2; they are shifted apart along the x-axis by the $\alpha$ parameter and converted into the polar space. During the next step in generating the structure each point location has a Gaussian kernel associated with it. As in the software retina, these kernels are parameterized by the density of neighbouring points to ensure a smooth and continuous representation of the visual signal; however, unlike in the software retina they are not used to subsample from an image but rather to project the contents of the image vector onto the image plane [1]. Finally, the two sets of point locations are rotated and aligned along their foveal sides so as to minimize the amount of empty space present in the resulting image.

$$Y_{cort} = \sqrt{(x+\alpha)^2 + y^2} \tag{2.1}$$

$$X_{cort} = tan^{-1}(\frac{y}{x+\alpha}) \tag{2.2}$$

Figure 2.5 shows an example image produced by the cortical mapping from an input image previously shown in Figure 2.4. The resulting image is of a lower resolution than the input image, with the exact compression ratio depending on the size of the software retina and the parameterization of the cortical mapping. The compression of this image format stems from how a wide range of visual frequencies captured by the software retina is regularized and represented in a narrow frequency range by expanding the densely sampled foveal region and compressing the sparsely sampled peripheries.

In images generated using the cortical mapping, translations correspond to a combination of translations, rotations and scaling that depends on movement relative to the fixation point. Radial translations scale the object, while translations about the fixation point rotate it. This space-variant transformation possesses a number of hypothesized benefits to deep learning vision systems, one of which stems from the decorrelation of noise from stable features owing to this mapping being conformal [2]. A conformal mapping preserves the structure of corner features but does not guarantee the same for other features and noise. This means that, given a corner feature and a series of cortically mapped images sampled by a retina fixated on different points around it, the feature will be represented in a consistent manner but the noise around it will be transformed inconsistently. This decorrelation is hypothesized to accelerate neural network training by helping the network focus on stable, corner-like features instead of overfitting to noise. The cortical transform also effectively augments training data and promotes scale and rotation invariance, as an object looks drastically different in the cortical image depending on its position relative to the fixation point. Figure 2.5 demonstrates this effect by transforming the image shown in Figure 2.4.

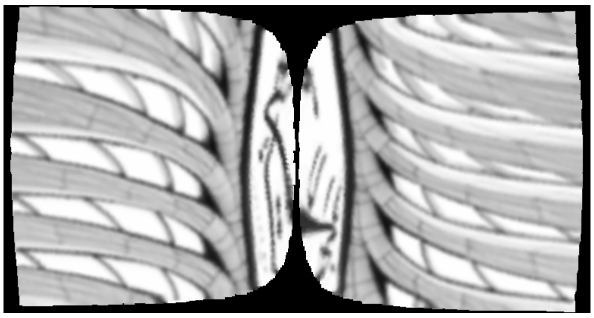


Figure 2.5: A cortical image produced from the input image shown at the top of Figure 2.4. Not to scale. Taken from [1].

The cortical mapping algorithm has undergone a pilot study in [1], where it was evaluated against a carefully selected subset of the Imagenet dataset [30] on a simple convolutional architecture. The purpose of that study was to demonstrate a proof of concept integration between the software retina and convolutional neural networks, which explains why the authors did not attempt to elaborate upon or investigate the mapping's hypothesized benefits. The experiments in [1] showed that the mapping enabled using the retina with convolutional neural networks with a moderate reduction in their image classification performance. For this reason during the first year of the project a more thorough investigation of the cortical mapping was considered as a possible avenue for research; however, it was not pursued as there were no ideas for meaningful and novel improvements to the mapping. Another possible research objective was to investigate a neural network layer that is capable of extracting visual features from the image vector directly, bypassing the need to generate a convolution-compatible image. Unfortunately, no such approach was found before the project's objectives have been revised.

## 2.3 Functional Expansions of the Retina

The first actioned research objective of this project was an expansion of the software retina's functional capabilities with a colour opponency model as well as a reproduction of the retinal scale-space pyramid from [3]. The goal was to investigate these and other retinal functions for any benefits to deep learning vision systems. In spite of these algorithms being investigated in a limited fashion, they have been fully implemented as they were being worked on in tandem with the specification of the retina experiments. This section details the algorithms that have been used to expand the functional capabilities of the software retina together with their corresponding limited investigations.

### 2.3.1 Scale Space Pyramid

The retinal scale-space pyramid was described by [3] and has been fully reproduced for this project's purposes. The algorithm includes a Gaussian pyramid and a Difference-of-Gaussians pyramid; both of these utilize successively coarser retina structures overlaid on top of each other, with each structure subsampling directly from the image vector of the structure below it. Including the retina structure that subsamples from the image, there are 4 levels in these pyramids. Each node in the successive pyramid levels has a Gaussian kernel associated with it, with the $\sigma$ parameter being scaled by the local node density as measured by the mean distance to the 5 closest neighbours within the same level. The field-of-view of each receptive field defines the maximum distance for subsampling from nodes in the lower level retina structure, and it is defined as $2.4 * \sigma$. The Difference-of-Gaussians retinal pyramid is computed by first computing two different retinal pyramids, one with narrow and one with wide receptive fields as defined by the ratio between their base $\sigma$ values, and then subtracting their image vectors from each other.

All data subsampled by the 4 levels of retina structures is stored in an image vector format, meaning that no backprojection algorithms that cast the data back to image space are necessary to compute the pyramid and all computations are performed directly on the image vector. A projection algorithm can still be used to visualize the visual data captured by the retinal pyramid, as seen in Figure 2.6.

The lack of projection algorithms in the retinal pyramid combined with the compactness of the image vector representation causes the retinal pyramid to have a very low memory footprint, as the image vector format is a much more compact representation of visual data than the pixel space. After sampling an input 926x926*px* image, a 4 level retina pyramid stores only $50000 + 12500 + 3125 + 781 = 66,406$ values in the image vector representation. Projecting this representation would mean storing images containing $926 * 926 + 463 * 463 + 232 * 232 + 116 * 116 = 1,139,125$ pixel values. In spite of this compactness, it is not obvious how would one go about integrating the pyramid with deep learning vision systems. The cortical mapping described in Section 2.2.3 could be used as an integration with convolutional neural networks, but using it would mean forfeiting the memory savings stemming from a compact image vector representation. A more efficient integration was sought that would enable a neural network to efficiently compute local features directly within the pyramid, but it was not found in time before the project's goals were revised.

The motivation behind implementing the image pyramid was that it would allow for the utilization of conventional computer vision techniques that operate in the more compact and efficient pyramidal scale space and, where applicable, enable a comparison of their efficacy against functionally analogous biomimetic algorithms. Had a more efficient interface with

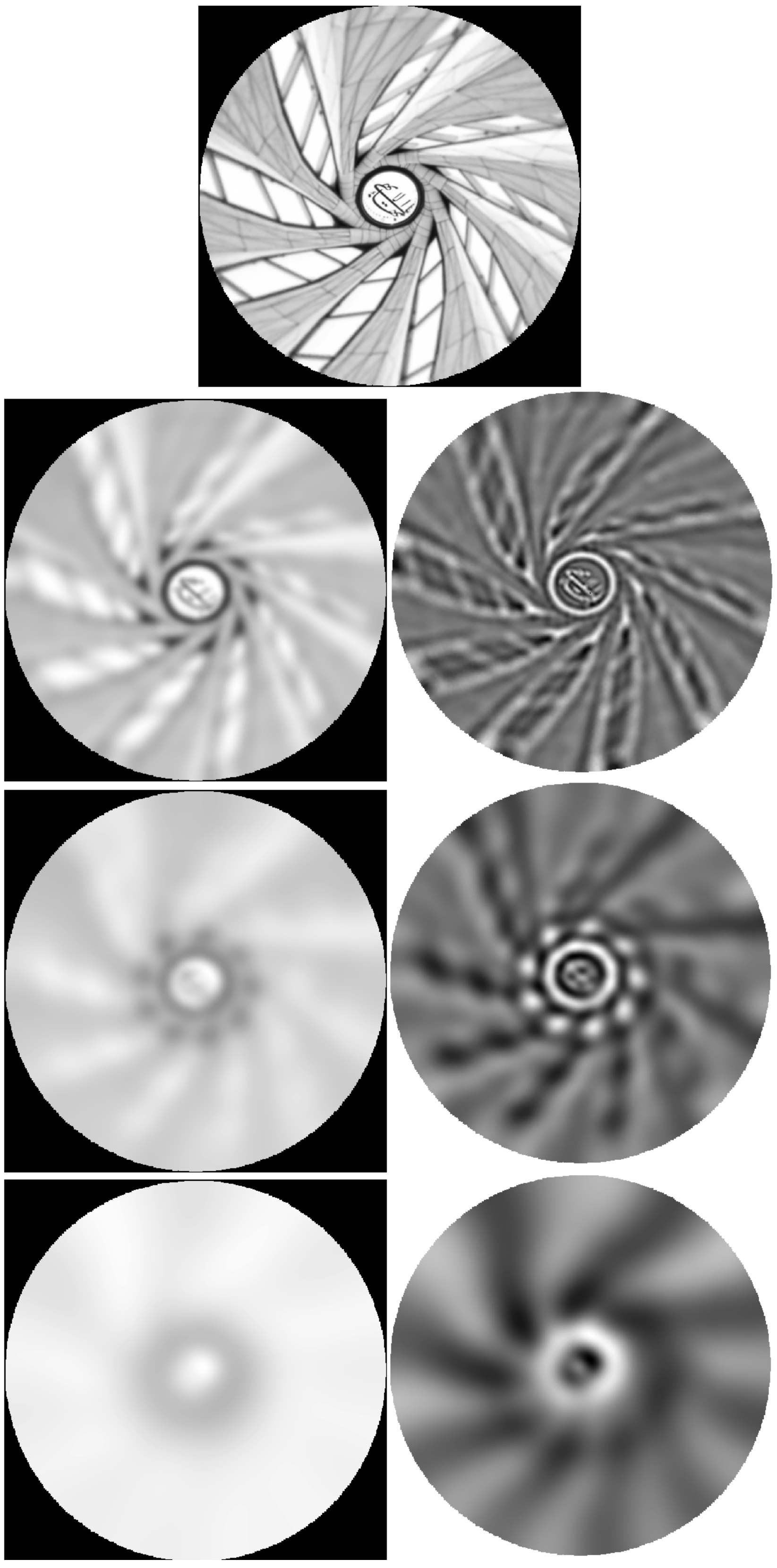


Figure 2.6: **Top:** the source image. **Bottom:** backprojections of the resultant Gaussian (left column) and Difference-of-Gaussians (right column) retinal pyramids.

neural networks been devised, the pyramid would have become a part of the attention module in the EPIC Kitchens investigations described in Section 2.4. The research plan included integrating the retinal pyramid into the optical flow estimating neural architecture defined in [31]) and feeding its output into the model's attention module. This was hypothesized to enable the module to compensate for global head movements in egocentric vision settings and to differentiate them from object motion and retina re-fixations. This plan was never realized; the amount of work required to achieve this and other research goals was severely underestimated and the EPIC Kitchens investigations were cut short before the retinal pyramid could be utilized or investigated.

### 2.3.2 Colour Opponency Model

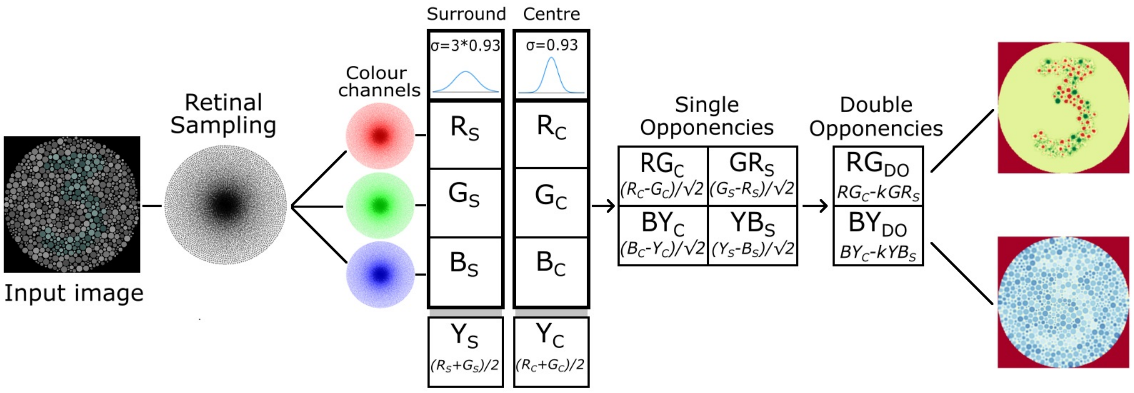


Figure 2.7: The single and double opponent retina cell model.

The colour opponency model introduced in this project is a functional model of single and double-opponent retina cells described in section 2.2.1. It was described in [32] and is based on the work of [33]. The motivating goal behind the model was to investigate whether a space-variant colour space inspired by the human retina can benefit vision systems by promoting invariance to illumination changes, colour constancy and improved colour contrast. In this model the image pixels are treated as signals coming from the retina's cone photoreceptors, the software retina's receptive fields are treated as the intermediate neurons in the retina that facilitate colour opponency, and the opponent image vector outputs are treated as retinal ganglion and lateral geniculate nucleus cells, with each image vector channel corresponding to a specific colour opponent cell species.

The algorithm implements the Marr-Hildreth difference of Gaussians operator [34] using two separate retina structures: one implementing the centre Gaussian receptive field with $\sigma = 0.93$ and another one implementing the surround Gaussian receptive field with $\sigma = 3 * 0.93$. The value of 3 scaling the surround Gaussian was chosen based on physiological findings regarding the structure of cat retinal ganglion cells [35]. The centre and surround retina

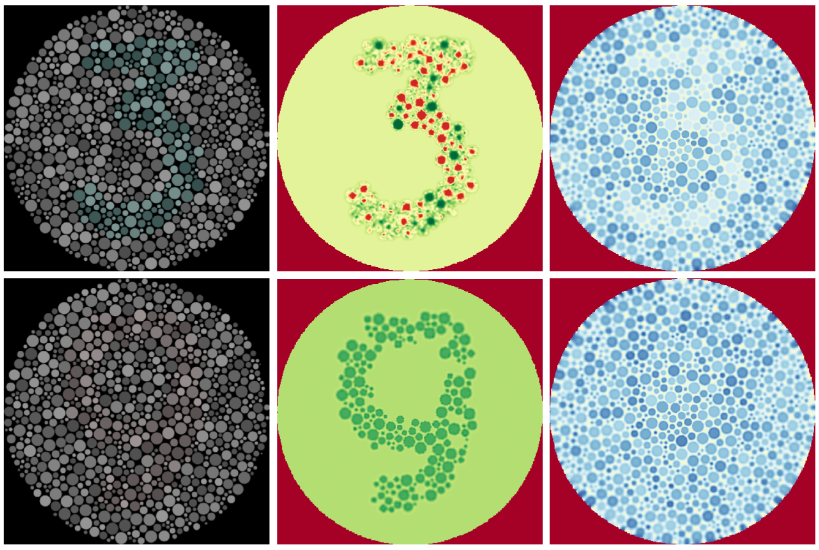


Figure 2.8: Low contrast colour-blindness tests processed using the double-opponency model. **Left:** Original images. **centre:** Retinal backprojections of the red-green double opponent cells, coloured using a divergent colour map. Red indicates negative values, yellow indicates values near zero, and green stands for positive values. **Right:** retinal backprojections of the blue-yellow double opponent cells, coloured using a divergent colour map. Red indicates negative values, yellow indicates values near zero, and blue stands for positive values.

structures are each used to subsample from the input image after the algorithm supplements the image's RGB colour space with a yellow channel $y = ((r + g)/2)$, resulting in a total of 8 image vector channels. These colour channels are used to model the outputs of four type-2 single opponent cell species: centre opponent r-g and b-y cells, and surround opponent g-r and y-b cells. The single opponent outputs are in turn added together in a spatially opponent manner to simulate two double opponent cell species: the red-green and the blue-yellow cells. The relevant formulas can be found in Figure 2.7.

The centre receptive fields of in vivo type 2 single opponent cells are not perfectly balanced with their surround receptive fields [24]. For this reason absolute stimuli generate a response from the cell, guaranteeing that their encoding together with the differential stimuli of the antagonistic fields. This feature is implemented by scaling the magnitude of the surround receptive field's response by k = 0.9 [33].

To qualitatively evaluate whether the colour opponency model can improve colour constancy a set of Ishihara tests commonly used for identifying colour blindness [36] was provided as input. Visualizing the modelled outputs of the double opponent cells has resulted in images with an improved colour contrast which is especially apparent in the two most challenging test images found, shown in Figure 2.8. The plan for quantitative evaluations was to investigate whether a deep learning vision system would benefit from using the simulated double opponent cells' output in classification tasks where colour constancy is a controlled variable used to increase task difficulty. The planned means of controlling colour was to use a synthetic dataset where illumination parameters as well as the colours of the classified object and the scene could be manipulated at will.

As mentioned before, the research objective of widening the functional capabilities of the software retina was set aside before these investigations could be completed. This research goal was deemed premature as no adequate experimental environment for rigorously evaluating the retina's impact on deep learning vision systems could be found in the literature. In addition to this, the biomimetic motives behind the colour opponency model were questioned by several publications. The findings in [37] [38] show that colour opponency features emerge spontaneously in convolutional neural networks, and that hardcoding them inside the retina would be redundant. In addition to this, [39] showed that colour processing is severely limited at the peripheries of human vision in a way that complicates our understanding of what is biomimetic colour processing; in their study subjects wearing VR headsets did not notice a complete loss of colour in most of their field-of-view.

## 2.4 EPIC Kitchens Investigations

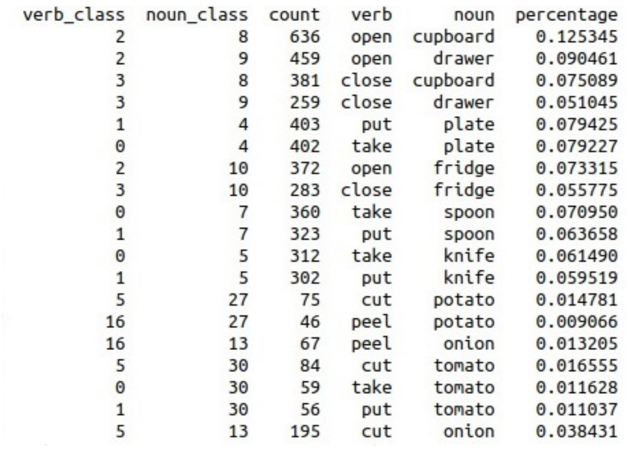
The lack of an adequate experimental environment for a rigorous evaluation of the software retina in deep learning systems was identified as a critical knowledge gap; to address this the focus of this thesis was shifted towards establishing such an experimental environment. This section describes the devised research framework, the EPIC Kitchens dataset [40], the motivations for using this dataset in investigating the software retina, and an architecture inspired by the Recurrent Attention Model (RAM) [10] [11]; it also explains the reasons why the research framework utilizing EPIC Kitchens was proposed, why it was set aside and what has been learned from it.

### 2.4.1 EPIC Kitchens and the Selected Subset

A major objective guiding dataset choice was to devise a long-lasting and rigorous testbed for a large number of potential future investigations of the software retina. As described in Section 2.2.1, many of the computations that take place in biological retinas process motion and temporal signals and reimplementing those computations within the software retina was considered for future work. For this reason a video dataset that can enable extracting temporal features was preferred to image datasets. The software retina's and cortical mapping's compression ratios scale with the retina's size, so a dataset consisting of high resolution images that could support a large retina was also preferred in testing the retina's memory efficiency. Large image size also gives the retina more room for exploring the scene, while high resolution visual features can only be resolved if the retina fixates on the appropriate location and thus support testing how well can a model control the retina. A rich feature space, such as one present in natural images, would also improve the sensitivity of any tests for the hypothesized scale and rotation invariant properties of the retina. Ideally, the structure of the visual task associated with the dataset would also require the architecture to locate a specific object within the scene as opposed to classifying the scene as a whole as that would further support evaluating the fixations produced by the active vision system.

At the time of its publication the EPIC Kitchens dataset was the largest egocentric vision dataset for action recognition from videos [40]. It consists of 55 hours of non-scripted recordings of the participants' daily activities in 32 different kitchens. The videos were recorded at a 1080p resolution with a GoPro camera mounted on the heads of the participants. All videos are broken up into action segments, with each segment being labelled with a verb and noun pair such as 'cut onion' or 'open cupboard'. At the time of utilizing it the dataset boasted a total of 125 verb classes, 331 noun classes and 39,594 action segments; however, newer versions have different statistics.

Although the EPIC Kitchens dataset fulfilled all of the aforementioned requirements, its size posed a significant challenge that could adversely impact the pace of the project. In response to this a subset of the dataset was selected, enabling faster architecture development and an easier interpretation of the model's behaviour. All of the problems related to the dataset's format and the architecture's development could be solved more quickly on a smaller subset, and the resulting deep learning architecture could be scaled up by increasing its depth and width if an evaluation on the full dataset was deemed necessary.



| verb_class | noun_class | count | verb | noun | percentage |
|---|---|---|---|---|---|
| 2 | 8 | 636 | open | cupboard | 0.125345 |
| 2 | 9 | 459 | open | drawer | 0.090461 |
| 3 | 8 | 381 | close | cupboard | 0.075089 |
| 3 | 9 | 259 | close | drawer | 0.051045 |
| 1 | 4 | 403 | put | plate | 0.079425 |
| 0 | 4 | 402 | take | plate | 0.079227 |
| 2 | 10 | 372 | open | fridge | 0.073315 |
| 3 | 10 | 283 | close | fridge | 0.055775 |
| 0 | 7 | 360 | take | spoon | 0.070950 |
| 1 | 7 | 323 | put | spoon | 0.063658 |
| 0 | 5 | 312 | take | knife | 0.061490 |
| 1 | 5 | 302 | put | knife | 0.059519 |
| 5 | 27 | 75 | cut | potato | 0.014781 |
| 16 | 27 | 46 | peel | potato | 0.009066 |
| 16 | 13 | 67 | peel | onion | 0.013205 |
| 5 | 30 | 84 | cut | tomato | 0.016555 |
| 0 | 30 | 59 | take | tomato | 0.011628 |
| 1 | 30 | 56 | put | tomato | 0.011037 |
| 5 | 13 | 195 | cut | onion | 0.038431 |

Figure 2.9: Statistics of the selected subset of EPIC Kitchens.

The selected subset consists of 5074 action segments with 19 unique activity classes made up of 6 verb and 9 noun classes. More detailed subset statistics can be found in Figure 2.9. There were several criteria involved in choosing the subset classes, the first of which was their incidence in the dataset as many classes in the dataset had only a handful of associated action segments. Some of the classes were selected to pose a requirement on the vision system of fixating the high resolution fovea on or close to the manipulated object; this included visually similar noun class pairs, such as potato vs onion, as well as verb classes that require differentiation between fine hand motions, such as peel vs cut. In order to provide an improved understanding of the tested architectures' temporal feature processing, verb classes were selected that are temporal inversions of each other, such as open vs close, and take vs put. The selected classes were split, with 20% of the samples going into the validation set and the remaining 80% going into the training set.

### 2.4.2 The EPIC-RAM Architecture

As the primary focus of the project at the time was the software retina, it was desirable to minimize the time spent on architecture search. Unfortunately no neural network architecture was found in the literature that could be used in these investigations without significant revisions, so an initial architecture was chosen with the intent of tailoring it to the project's needs. To be compatible with EPIC Kitchens the architecture had to be sufficiently large to be suited for working with visually rich natural scenes, and it had to be capable of processing videos while fixating at different locations throughout their duration. The latter requirement effectively eliminated common action-recognition architectures such as ResNet (2+1)D [41] from being integrated with the retina, as they relied on creating a tensor of stacked video frames and processing it in a passive manner using spatiotemporal convolutions. This requirement suggested recurrent architectures as likely candidates for investigating the software retina.

The software retina's exploration of the scene could be either driven by a hard-coded, standalone visual saliency algorithm, or it could be driven by an integral part of the neural network architecture. The latter approach was deemed more desirable as designing a separate visual saliency algorithm would require solving several challenging problems, one of which is implementing an inhibition of return[1] mechanism that differentiates between and accounts for different sources of optical flow present in egocentric videos. In addition to this, the human vision system is said to be closely integrated with other neural functions [43] making an integrated solution not only more practical but also more biomimetically motivated.

The architecture selected to be the starting point for further development was the Recurrent Attention Model [10] [11], hereby abbreviated as *RAM*. It uses an Elman RNN to aggregate the visual information collected during multiple observations of the scene. To subsample from the input image RAM uses a crude foveated sensor that crops out square, variable resolution image patches. Visual features are extracted from these patches and combined with the current fixation coordinates. This output is fed into the RNN which maintains a

[1]Inhibition of return is a mechanism for preventing the active vision system from repeatedly refixating on a recently attended part of the scene. [42]

hidden state representing the network's understanding of its interaction with the scene. The hidden state is used by 3 other networks to perform their respective functions: the locator network parameterizes a probability distribution used to sample the coordinates of the next fixation, the baseline network predicts the future reward of the locator network in order to regularize its reward, and the action network outputs the predicted classification once a pre-defined number of observations have been subsampled from the scene. The locator network is trained solely using reinforcement learning, with the reward being regularized using the baseline network's prediction. The rest of the RAM architecture is trained solely using gradients from the negative log-likelihood classification loss.

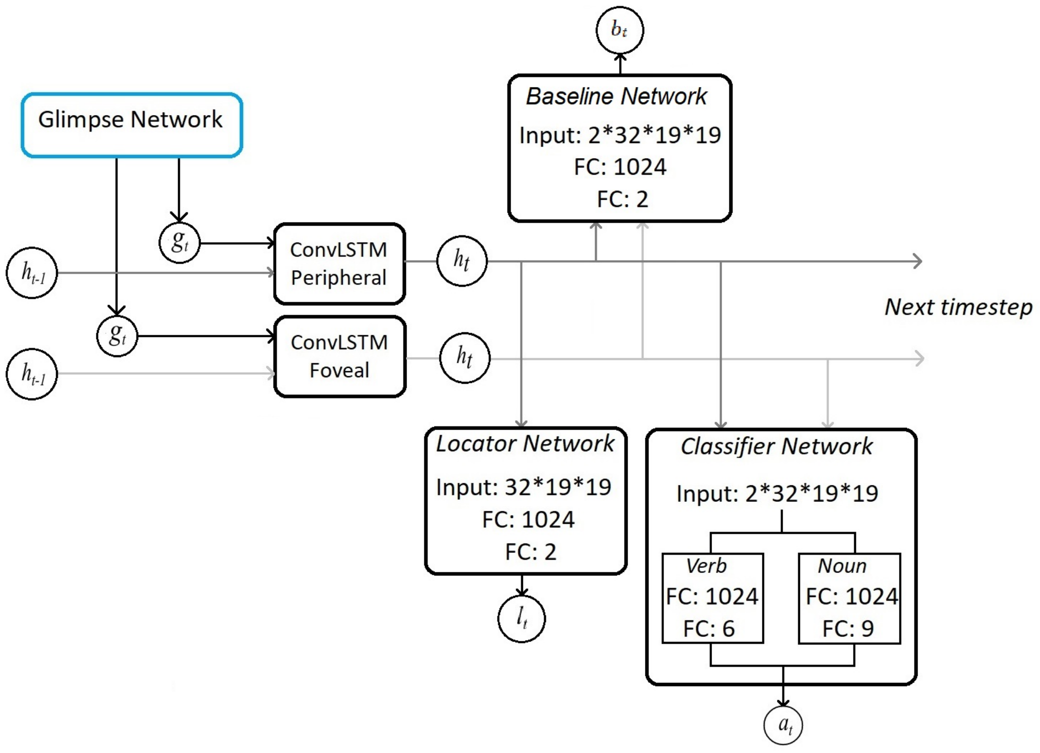


Figure 2.10: An overview of the EPIC-RAM architecture.

The RAM architecture fulfils all of the functional requirements needed to process videos using the software retina. The crudely foveated image patches serving as its sensor can be replaced with the software retina, and the RNN enables it to process sequences of video frames. The location module is also a simple and integrated way for the architecture to select new fixation locations. However, the RAM architecture has been originally developed for and investigated on quite simple 'toy' challenges such as the MNIST dataset [44] and the SVHN dataset [45]. As a result it is not suitable out-of-the-box for processing high resolution and visually rich real-world scenes.

The working name of the RAM architecture that has been adapted for the purposes of this project is EPIC-RAM. Its high-level function is the same as that of RAM; however, it has undergone a number of significant revisions. First, the resolution of RAM's sensor patches was increased to 50x50*px* for the foveal patch and 100x100*px* downscaled to 50x50*px* for the peripheral patch. This was done alongside downscaling the input images from 1920x1080*px* to 456x256*px* in order to increase the field-of-view of the vision system and enabled it to capture a sufficiently large part of the activity in its high resolution foveal patch. These sensor and input parameters were selected to simplify and speed up the architecture's development; the plan was to parameterize the patch sensor to have a comparable field-of-view with that of the software retina during its evaluation once the architecture is complete.

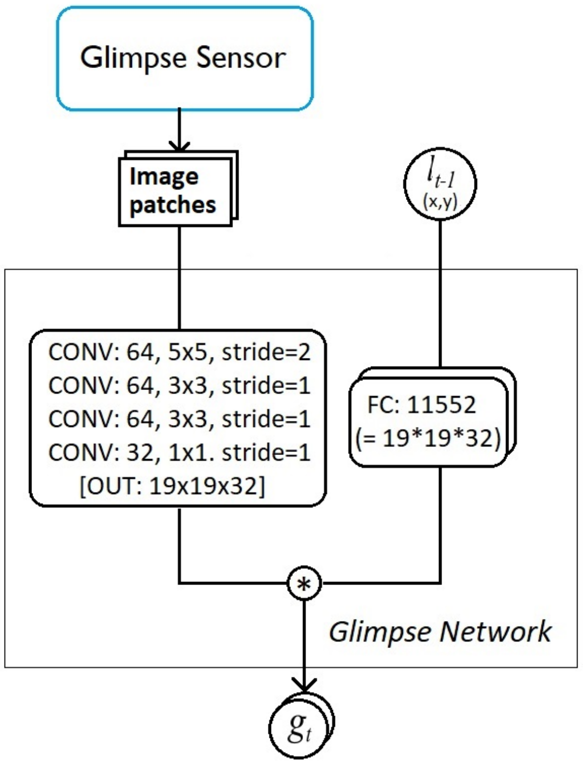


Figure 2.11: EPIC-RAM's feature extraction network, dubbed the glimpse network, that produces a feature vector $g_t$ which combines both visual and proprioceptive information.

A high-level overview of the EPIC-RAM architecture can be found in Figure 2.10. Borrowing the terminology used by the authors of the RAM architecture, a singular observation of the environment is referred to as a glimpse. A *glimpse sensor* is the algorithm used to sample from the image, which in this case refers to cropping out overlapping image patches. The *glimpse network* is a feature extraction network that combines visual features with proprioceptive information and outputs the glimpse vector ($g_t$), which contains all of the features extracted by the glimpse network at the current timestep. Given that the glimpse sensor outputs two images - a foveal and a peripheral image patch - two glimpse vectors are produced. The fully connected layers that the RAM glimpse network uses to extract visual features from image patches have been replaced by convolutional layers. This was necessary as fully connected layers are not normally suitable for feature extraction from large image inputs due to difficulty with training and their parameter cost. The adapted feature extraction architecture can be seen in Figure 2.11 The Elman RNN at the heart of the RAM architecture has also been replaced with a convolutional LSTM [46] as processing the output of the feature extraction module with a fully connected layer would cause a significant parameter overhead. In Figure 2.10 $h_t$ refers to the LSTM's hidden state at the current timestep, $h_{t-1}$ refers to the previous timestep's hidden state, $l_t$ refers to the fixation coordinates chosen for the next observation and $a_t$ refers to the model's classification output. To account for two component labels of the activity classes consisting of a noun and a verb, the reinforcement learning reward used to train the locator network was set to 0.5 for each correct label component.

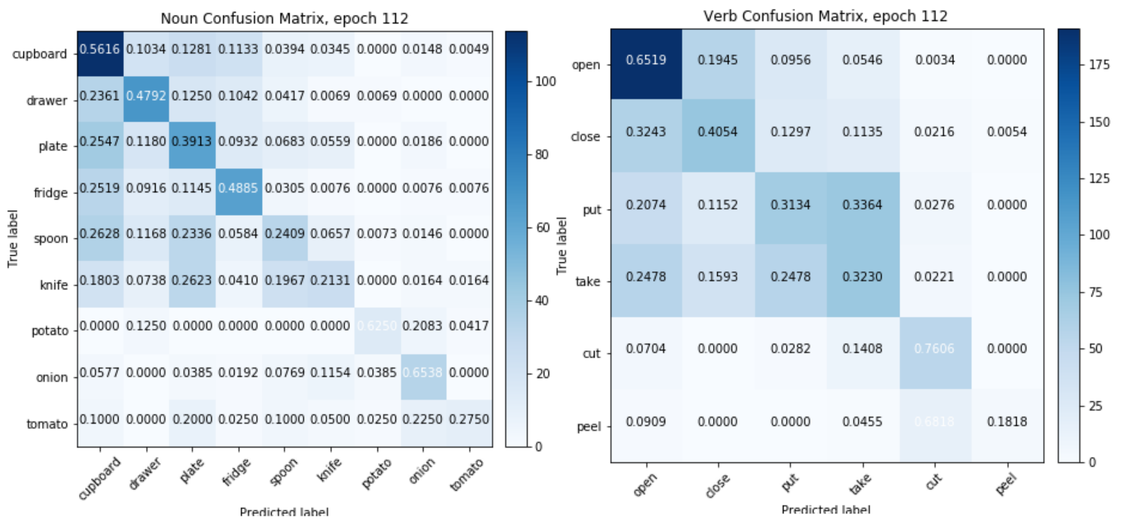
Finally, inspired by the DRAM architecture [11], the EPIC-RAM architecture was modified to utilize a two-stream approach, where the two image patches pass through the glimpse network separately to produce two feature tensors that are passed to separate convolutional LSTMs. The peripheral LSTM's hidden state is passed to the locator, baseline and the classifier networks, whereas the foveal LSTM's hidden state is only passed to the classifier and the baseline networks. The motivation behind this design choice is that the problem of merging foveal and peripheral features is not trivial, and the locator network should only need peripheral data in order to decide where to fixate next.

### 2.4.3 Experiments

As the planned investigations were going to utilize the selected subset of EPIC Kitchens it was necessary to use that subset to evaluate a benchmark architecture that can act as a reference point. This could also help identify and address any detrimental aspects of the EPIC-RAM architecture that could confound retina evaluations. The selected benchmark architecture is the R(2+1)D model [41], which is a variant of the popular ResNet architecture [47] that has been tailored for action recognition. It is a passive vision system, so it cannot be integrated with the software retina.

Both the R(2+1)D and the EPIC-RAM architectures were trained using the AdamW optimizer [48] with a learning rate of 0.003. The loss used in classification was the negative log-likelihood loss. Early stopping was employed to end the training process if validation accuracy did not significantly improve for over 15 epochs. The input videos were spatially downscaled to 456x256*px* and, in order to address excessively long action segments, they were also temporally subsampled by a factor of 5 down to a maximum of 50 frames. Due to memory constraints and other technical issues associated with the processing of variable length sequences by a recurrent network the batch size was kept at 1 during both training and validation.

Figures 2.12 and 2.13 show the results of training the EPIC-RAM architecture on the subset, whereas Figures 2.14 and 2.15 show the results achieved by the benchmark architecture.



| | cupboard | drawer | plate | fridge | spoon | knife | potato | onion | tomato |
|---|---|---|---|---|---|---|---|---|---|
| cupboard | 0.5616 | 0.1034 | 0.1281 | 0.1133 | 0.0394 | 0.0345 | 0.0000 | 0.0148 | 0.0049 |
| drawer | 0.2361 | 0.4792 | 0.1250 | 0.1042 | 0.0417 | 0.0069 | 0.0069 | 0.0000 | 0.0000 |
| plate | 0.2547 | 0.1180 | 0.3913 | 0.0932 | 0.0683 | 0.0559 | 0.0000 | 0.0186 | 0.0000 |
| fridge | 0.2519 | 0.0916 | 0.1145 | 0.4885 | 0.0305 | 0.0076 | 0.0000 | 0.0076 | 0.0076 |
| spoon | 0.2628 | 0.1168 | 0.2336 | 0.0584 | 0.2409 | 0.0657 | 0.0073 | 0.0146 | 0.0000 |
| knife | 0.1803 | 0.0738 | 0.2623 | 0.0410 | 0.1967 | 0.2131 | 0.0000 | 0.0164 | 0.0164 |
| potato | 0.0000 | 0.1250 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.6250 | 0.2083 | 0.0417 |
| onion | 0.0577 | 0.0000 | 0.0385 | 0.0192 | 0.0769 | 0.1154 | 0.0385 | 0.6538 | 0.0000 |
| tomato | 0.1000 | 0.0000 | 0.2000 | 0.0250 | 0.1000 | 0.0500 | 0.0250 | 0.2250 | 0.2750 |

| | open | close | put | take | cut | peel |
|---|---|---|---|---|---|---|
| open | 0.6519 | 0.1945 | 0.0956 | 0.0546 | 0.0034 | 0.0000 |
| close | 0.3243 | 0.4054 | 0.1297 | 0.1135 | 0.0216 | 0.0054 |
| put | 0.2074 | 0.1152 | 0.3134 | 0.3364 | 0.0276 | 0.0000 |
| take | 0.2478 | 0.1593 | 0.2478 | 0.3230 | 0.0221 | 0.0000 |
| cut | 0.0704 | 0.0000 | 0.0282 | 0.1408 | 0.7606 | 0.0000 |
| peel | 0.0909 | 0.0000 | 0.0000 | 0.0455 | 0.6818 | 0.1818 |

Figure 2.12: Evaluation set confusion matrices from EPIC-RAM's final training epoch. The colours code for the absolute number of action segments. The values in the grid indicate the proportion of the true class' action segments predicted to belong to each listed class. Class labels are sorted by their incidence in the dataset.

These are only preliminary results that were meant to guide further refinements of the EPIC-RAM architecture in advance of integrating it with the software retina; however, further experiments were deemed unnecessary as the focus of the project has shifted.

The EPIC-RAM architecture has demonstrated meaningful learning by achieving 30% accuracy on the validation set. The resulting validation set confusion matrices in Figure 2.12 show that there was a significant degree of overfitting to the most prevalent classes in the subset, both in the noun and the verb domains. Aside from the overfitting, the verb confusion matrix shows that the model was easily confused between the verb class pairs that are temporal inversions of each other: 'open' vs 'close', and 'put' vs 'take'. The least prevalent verb class, 'peel', has been dismissed by the model almost entirely in favour of the visually similar 'cut'.

The benchmark R(2+1)D architecture has not demonstrated any meaningful learning; as seen in Figure 2.15 its validation loss has only increased throughout training and its validation set confusion matrices in Figure 2.14 show that this architecture has overfitted to the most prevalent classes. Although the failure of R(2+1)D and the relative success of EPIC-RAM can be interpreted to suggest that the selected subset is too challenging for the benchmark architecture and that the proposed network can tackle challenging visual tasks, these results are only preliminary and have not been further validated with a reproduction. It is worth noting that the restriction of batch size to 1 could have adversely affected R(2+1)D.

Training EPIC-RAM took several days due to the video format of the dataset, batch size being restricted to 1 and the training taking 112 epochs to complete. Integrating the software retina with the architecture was attempted, but the software retina increased the computa-

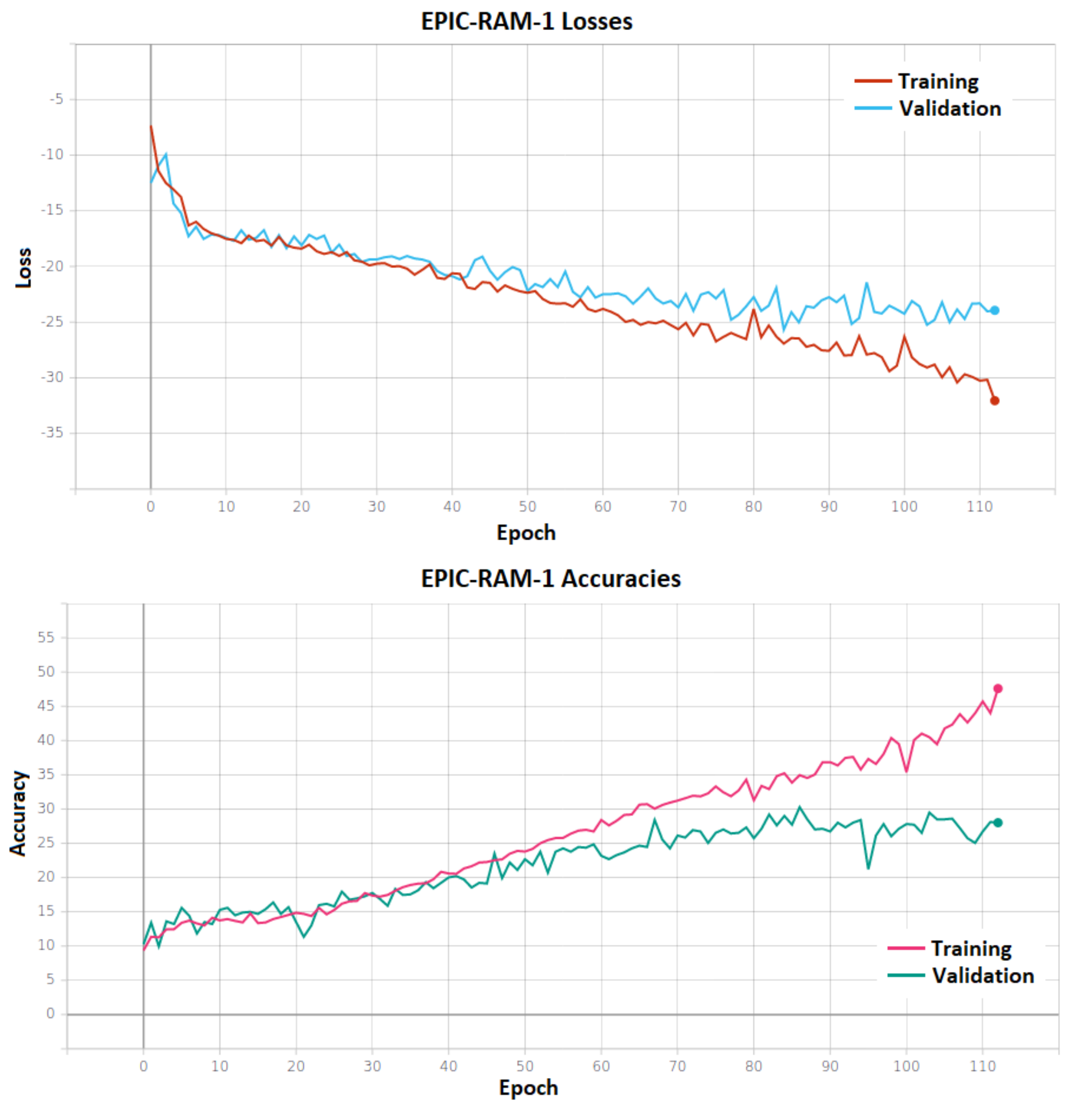


Figure 2.13: Accuracy and loss plots of the EPIC-RAM architecture.

tional cost of EPIC-RAM's forward pass and further slowed down training beyond a week. The lack of a ready, retina-compatible architecture has already resulted in delaying retina investigations. At this point the feasibility of pursuing the software retina was put into question, as resolving these challenges was necessary for a rigorous evaluation of the retina, yet it risked consuming too much time.

## 2.5 Discussion

This chapter presents the exploratory work that was carried out in the early stages of the project and in doing so it describes the motivations for suspending the pursuit of the software retina. Section 2.3 described expansions of the retina's functional capabilities and the underlying motivations, whereas Section 2.4 describes the development of an experimental

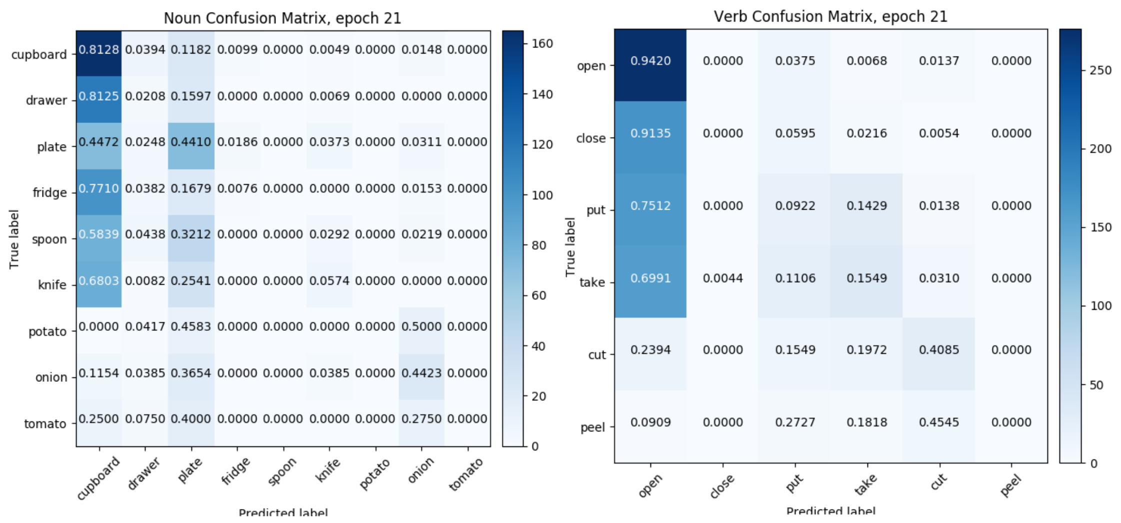
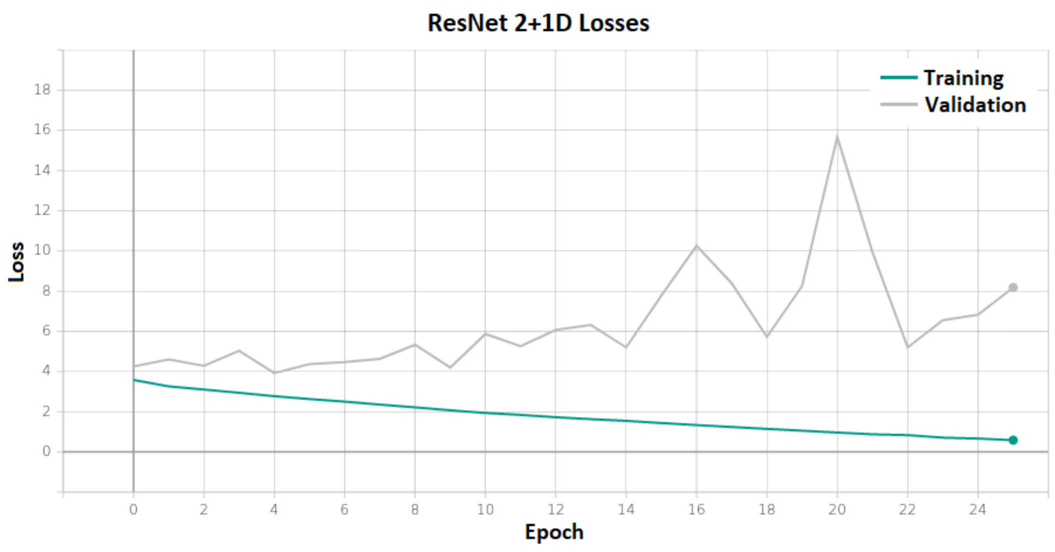


Figure 2.14: Evaluation set confusion matrices from R(2+1)D's final training epoch. The colours code for the absolute number of action segments. The values in the grid indicate the proportion of the true class' action segments predicted to belong to each listed class. Class labels are sorted by their incidence in the dataset.

framework for evaluating the software retina using the EPIC Kitchens dataset and a variant of the Recurrent Attention Model architecture. At its outset the project had a positive outlook on researching the software retina; it was incorrectly presumed that once the retina's functionalities are expanded it will be relatively straightforward to evaluate their contributions to deep learning vision.

At first, investigating the expanded functional capabilities of the retina was postponed until the completion of a retina-compatible deep learning environment. As this exploratory work progressed and as more challenges associated with pursuing the software retina were uncovered it gradually became apparent that investigating the retina cannot be tackled in a sufficiently rigorous manner within the time constraints of this project. The motivations behind evaluating colour opponency within the software retina were questioned by findings of emergent colour opponency in deep learning systems [37] [38] as well as complications in how the human vision system processes colour at the peripheries of its field-of-view [39]. The establishing of a rigorous research framework for evaluating the software retina and its expansions was found to be not feasible within the time constraints of this project; as a result investigating the software was abandoned entirely in favour of the active vision approach described in Section 3.4.

The work presented in this chapter was instrumental in defining a productive and actionable research programme that drove the investigations described in the following chapters. This work was thus formative to this project, in spite of not yielding convincing empirical data. Instead of advancing the initially posed research objectives this work questioned their feasibility and motivated their revision. It was learned that any meaningful evaluation of the



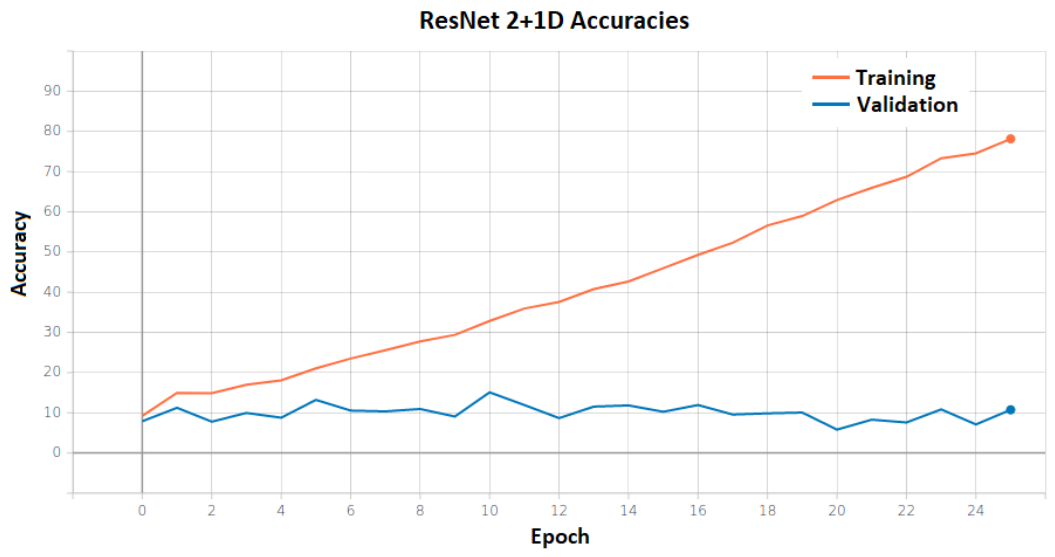


Figure 2.15: Accuracy and loss plots of the R(2+1)D architecture.

software retina's impact on deep learning vision systems requires a mature - meaning well understood and developed - active vision system together with established active vision research practices, and that these are currently lacking within deep learning literature. Section 3.4 describes the proposed research framework that serves to aid in studying and developing active vision systems.

# Chapter 3

# Active Vision: Literature & Proposed Framework

***Abstract:*** *Transitioning the focus of the thesis to active vision called for a new literature survey together with a re-formulation of the guiding framework and the main objectives. This chapter reviews the relevant active vision literature, uses several experiments to further comment on its state and then proposes and discusses a framework for understanding and studying active vision. When compiled together, the active vision literature reviewed in Section 3.2 can be seen to lack cohesion as many of the published works pursue disjointed approaches that do not reinforce each other. Section 3.3 describes experiments that demonstrate pressing research gaps pertaining to key issues in the active vision literature. The lack of cohesion and the research gaps are addressed in Section 3.4 with the proposal of an active vision framework alongside arguments for its use in directing and organising future active vision research.*

## 3.1 Motivation and Objectives

The overarching objective of this project is to further the understanding of active vision by developing and investigating a deep learning active vision architecture that can solve a relatively simple visual task on a non-trivial dataset. In spite of rarely explicitly framing the research in such terms, prior literature has already engaged with active vision in simplified settings or using toy datasets. The goal this chapter is to support the main goal of this project by clarifying the gap in the active vision literature and establishing a set of research practices to guide active vision investigations.

Although the research focus of the project has shifted away from the software retina, many of the underlying motivations have remained the same. The most significant motivation for the pursuit of active vision that persisted from the software retina investigation is that of biomimetics, or mimicking biological systems for the purpose of solving human problems. There are arguably no known examples of passive vision in nature, as all visually endowed animals actively explore their surroundings by reorienting their eyes, heads or entire bodies.[1] In contrast, the vast majority of deep learning vision approaches are passive. The author believes that this is a fundamental issue that misdirects the research efforts within deep learning to developing solutions that excel on highly controlled research datasets but struggle in less constrained, real-world settings. A field-wide switch to experimenting in real-world environments is one hypothetical solution to this issue; however, it would not be feasible as the associated costs could drastically reduce the volume of research performed. Active vision presents an opportunity for bringing the development setting of vision algorithms closer to the real-world setting; it puts forward a set of requirements on the vision system's function with the aim of relaxing constraints on the type of problems that the system can solve and on the environments that it can operate within.

## 3.2 Literature Review

### 3.2.1 Active Vision vs Visual Attention

A vision system is considered to be active if is capable of exploring its environment by controlling the geometric parameters of its visual sensor [4]. Passive vision systems do not have a narrow definition; they are simply said to be vision systems that are not active. Passive vision has been previously associated with Marr's paradigm of three stages of vision [50] due to its emphasis on low-level and bottom-up visual processes that fully elaborate the visual scene.

Within computer vision, the interest in active vision has been initiated by the publications of Aloimonos et. al. [4] and Ballard [9]. Aloimonos et. al. have motivated active vision by showing that various low-level visual problems, such as structure from motion, shape from shading, shape from texture and shape from contours are more tractable with an active observer capable of perceiving the object of interest from different viewpoints [4]. Ballard has argued for the benefits of active vision with foveated gaze control, one of which is that various vision tasks involving motion are simplified when using an exocentric coordinate frame, i.e. when the observer has locked their gaze onto a moving object to tracking it, which is equivalent to using a location on the object as the origin of the observer's coordinate space [9].

[1]The box jellyfish can be argued to rely on passive vision due to an eye type that looks upwards irrespective of the jellyfish's orientation; however, the animal possesses more eyes that are active and is still capable of exploring its environment by traversing it [49].

Not all neural vision systems can be done justice by merely shoehorning them into the active and passive categories. The concept of visual attention and the terminology that surrounds it can help differentiate the characteristics and behaviours of different vision systems. However, an overview of the literature on visual attention and active vision shows that it is not immediately obvious why some researchers choose one term over the other when describing their work. Jayaraman et al. [13] state that active vision aims to sample new data, whereas visual attention aims to selectively suppress or focus on parts of already available data. This distinction is somewhat accurate within the scope of deep learning literature, but it does not agree with prior psychophysics and vision literature which would define active vision as incorporating overt attention [51] [52] and the selective deployment of cognitive machinery within the current field of view as covert attention [53]. Covert attention does not involve visual sensor action, whereas overt attention does. Moreover, some deep learning publications such as that of Mnih et al. [10] describe active vision systems without ever referring to them with the term “active vision“, opting for “visual attention“ instead. It appears that active vision is the preferred term for researchers coming from a robotics background and for those who are more familiar with psychophysics literature from prior decades. It also appears that due to a recent surge in the number of highly impactful publications proposing covert attention approaches in deep learning, most notably Xu et al. [54], the deep learning community has abandoned the more accurate terminology in favour of understanding all attention to be covert.

Chun and Wolfe refer to attention as “a multifaceted term referring to a number of different acts and loci of selection“ [55]. This is accurate, as there are at least three distinctions that can help disambiguate visual attention. First, attentional processes can be divided into the pre-attentive and the attentive stages [55]. Pre-attentive processes process the full field of view in a highly parallelized manner, whereas attentive processes are more sequential and decide on the information to be further processed. The next distinction lies between exogenous and endogenous attention, which is also often termed bottom-up and top-down attention. Exogenous, or bottom-up, attention is guided by external stimuli, whereas endogenous, or top-down, attention is guided by a specific policy.

The final distinction lies between the aforementioned covert and overt attention. Overt attention is tied to the visual sensor and is thus constrained by its architecture. Covert attention is deployed internally and thus it suffers from no such constraints; it can take any, even disjointed, geometric form over the field of view. Within the scope of psychophysics covert attention can be understood as a strictly pre-attentive process whose primary role is to pre-process a peripheral area of the field of view before it is overtly attended by our gaze [53].

### 3.2.2 Recurrent models of visual attention

The literature focused on the Recurrent Attention Model (RAM) architecture is the most widely cited active vision literature in deep learning, in spite of most of it not using the term “active vision“. In their seminal work, Mnih et. al. [10] have introduced the RAM architecture that interacts with input images via a virtual, foveated, bandwidth-limited sensor to classify hand-written digits in the MNIST dataset [56]. When given a specific image location, the sensor extracts overlapping image patches at various resolutions and passes them on to a glimpse network that serves the function of a feature extractor. The extracted features are then passed to a recurrent neural network (RNN) which uses them to update its internal hidden state, i.e. memory. The hidden state is the input of a location network whose output determines the next location to be observed by the sensor. The whole process repeats for a pre-specified number of iterations after which the hidden state is passed to a network that outputs the final classification of the input image. The location network is trained using reinforcement learning, whereas the rest of the model is trained using classification gradients.

This architectural design consisting of a feature extractor with a restricted field of view feeding features into a recurrent network whose hidden state is used for downstream tasks was adopted in numerous later publications. In a follow-up investigation, Ba et al. [11] extend the RAM architecture to introduce the Deep Recurrent Attention Model (DRAM) that was trained to read house numbers on the SVHN dataset [45]. Their main extension to RAM was the introduction of two parallel processing streams, one for the foveal image patch and one for the peripheral image patch, each with its own recurrent memory network. Other significant modifications consisted of replacing the fully connected RNN that formed RAM’s memory with a pair of LSTM networks [57] and introducing a context network that processes a low-resolution view of the full image to initialize one of the LSTMs’ hidden state. By having access to the entirety of the input image, the context network utilizes passive vision making the DRAM architecture violate the restrictions and principles of active vision. Although strictly speaking DRAM is not an active vision architecture, both the novelties it introduces as well as its means of incorporating passive vision exemplify some of the problems inherent to active vision; namely its sensitivity to starting conditions and the different functional utilities of the visual data captured in the foveal and in the peripheral regions of the sensor.

Further building on the DRAM architecture, Sermanet et al. [15] replace DRAM’s simple feature extractor with a pretrained GoogLeNet [58] and train it on the Stanford Dogs dataset [59]. Similarly to the DRAM architecture, this model bypasses the active visual sensor to uniformly scan the full image to choose the first fixation location and thus incorporates passive vision. Additionally, the sensor used in this work has a really wide field of view, with the lowest resolution image patch being as large as the input image and the half-resolution patch being half as large. Relaxing the constraints imposed on the vision task in such a manner is effective if the goal is to maximize the model's performance; however, it is not valuable in the pursuit of active vision and learning where to look as it unrealistically trivializes the problems inherent to active vision instead of solving them. The large field of view has also caused their model to gain only negligible improvements in performance from making multiple observations of the input image, raising the question of why have the authors chosen to pursue an architecture designed for processing multiple foveated observations in the first place.

In a functional expansion on the RAM architecture, Li et al. [12] have accelerated their architecture's average processing time by introducing a module that enables it to decide whether it should sample another observation from the image or complete its interaction by outputting the predicted classification label. They evaluated their model on Stanford Cars [60] and the CUB-200-2011 [61] datasets. Although they only reported a reduction in processing time and do not report any gains in accuracy, their work describes and investigates a number of training methods that can help tackle the problems associated with the RAM architectures.

The work of Jayaraman et al. [13] [14] has introduced an unsupervised source of gradients into an adapted version of the RAM architecture in the form of a LookAhead module. This module is trained to predict the next hidden state of the RNN conditional on the current hidden state, current sensor pose and next sensor motion. The authors hypothesized that this improves their model's classification performance by promoting the learning of stable features that respond in a learnable and systematic way to sensor motion, but in spite of achieving marginally improved accuracy on scene and object recognition tasks they showed no evidence in support of this hypothesis. It is worth noting that these two publications are the first in the research thread building on the RAM architecture to describe their problem domain with the term "active vision".

Although Cheng et al. [16] only gave scant mention to RAM in their work, their model broadly follows the same architectural design. Just like Jayaraman et al. above, they referred to their work with the term "active vision". Unlike prior iterations of the RAM architecture which primarily focused on 2D environments, their model is designed for exploring 3D environments. They introduced a geometry-aware recurrent network that extracts depth and foreground object masks from RGB images, unprojects them together with the RGB image into a 3D feature tensor and then transforms this tensor to align it with the model's first observation and use it as input to a 3D convolutional GRU memory [62]. The GRU's hidden state is then fed into an encoder-decoder network to produce the model's predicted reconstruction, segmentation and classification. The policy network that controls the active camera is trained used reinforcement learning and takes as input the GRU's hidden state alongside the input RGB image. Although this work did not present any substantially novel solutions to the problem of overt attention in active vision, it introduced an elaborate functional expansion to the architecture's recurrent memory module, enabling it to aggregate observations from multiple points in 3D space. This model's use of the unprojection mechanism alongside the 3D feature transformation demonstrates the need for active vision systems to align and spatially 'stitch together' the information acquired during multiple observations from different parts of the scene.

### 3.2.3 Datasets and Reinforcement Learning for Active Vision

Although not all literature on active vision in deep learning builds on the RAM architecture, the vast majority of it also adopts some form of reinforcement learning to solve the problem of choosing where to look. Ammirato et al. [6] produced a dataset that emulates an indoor active vision setting and used reinforcement learning to train an agent that gradually refined its viewing position and orientation in order to optimize its classification performance. Their model does not accumulate evidence across multiple observations; instead it chooses discrete actions that reorient and move the sensor to produce a more valuable observation that leads to a better classification performance. The dataset introduced in their work only allows the model to perform one discrete action from a small pre-defined set, thus restricting the model from learning to freely explore its environment.

Malmir et al. [63] [64] have also introduced a dataset emulating active vision and trained a reinforcement learning agent to solve its object recognition task. Their GERMS dataset consists of video recordings of a robot performing give-and-take trials where the robot picks up an object, rotates it to examine it, and returns it; it amounts to a viewsphere dataset with each object having 6 grasping orientations viewed from a range of angles resulting from being moved towards the robot's camera and rotated 180 degrees. As a result, just like with the dataset introduced by Ammirato et al., any model trained using this dataset will be restricted to taking discrete actions instead of freely exploring a continuous scene. Such a constrained mode of exploration is a significant limitation as it can prevent a system with a narrow fovea from fixating on a key feature in the scene.

In other related works Gartner et al. [65] have used reinforcement learning to train an agent to estimate the pose of humans standing within a dense camera rig. The rig amounts to a viewsphere and it enables the agent to choose from 30 different viewpoints. Their model fuses the multiple observations together, but it does not utilize a recurrent network. Cheng et al. [66] utilize reinforcement learning to teach a robot to deal with occlusions by jointly controlling its gripper and camera. They test multiple actor-critic architectures and use a region proposal network to choose where to look, which is an inefficient and brute-force-like approach when contrasted with the recurrent models of visual attention described in Section 3.2.2 that utilize their maintained memory state in choosing where to look. Mathe et al. [67]

use a sequential reinforcement learning model for object detection, but both their setting and architecture trivialize active vision. Their model has perfect memory, meaning it has unrestricted access to its past observations. It also only selects out of a set of pre-proposed image regions and once it fixates within one it adds the whole region to its memory.

### 3.2.4 Embodied AI

Just like active vision imposes its own set of problems and constraints on passive vision, embodied AI complicates machine learning by imposing its own constraints on the learning process. Smith et al. [68] introduced the embodiment hypothesis which draws facts from human babies' development to argue that intelligence emerges during sensorimotor interaction of an agent with its environment and that grounding the agent in a coherent and multi-domain world is crucial for developing human-like intelligence. This concept is closely related to active vision, to the point that embodied AI agents often utilize active vision when visually exploring their environments at training time.

Yang et al. [69] introduce the task of embodied visual recognition, where an agent can navigate a virtual 3D environment to perform a number of visual tasks. They develop the Embodied Mask-RCNN architecture that enables the agent to move strategically in a way that improves the process by which it learn visual recognition. In embodied AI agents always operate in 3D environments, virtual or real-world, both at training and at test time, and tend to make decisions solely based on egocentric perceptual inputs. As a result the structure of the agent's training phase becomes non-trivial as the agent now has to choose how to explore its environment for learning purposes. This too is a case of active vision; however, here it is being used to a different end than in the literature reviewed above.

Chaplot et al. [70] use active vision to devise an exploration policy for visual learning that rewards mistakes; their method searches for areas that, when explored from different viewpoints, lead to inconsistent predictions in order to maximize the useful corrective gradients in the object detector. This approach is yet another way in which active vision can be utilized at training time, but for this reason it is not particularly relevant to this thesis.

### 3.2.5 Active Inference and the Free Energy Principle

The free energy principle is an attempt at explaining how adapting systems such as the brain resist collapsing into disorder; it states that all self-organizing systems must minimize their free energy in order to remain at equilibrium with their environments and to prevent collapsing into disorder [71]. Within the context of the brain, free energy is defined as an information theory measure that is an upper bound on the surprise coming from sampling data [72]. Active inference is an application of the free energy principle to the brain and to artificial neural networks; it postulates that the brain maintains a generative model of the world that can predict the sensory data that it will receive in the future [73]. The relative popularity of this theory has motivated some deep learning researchers to apply it to their work. In their paper, the authors of the aforementioned DRAM architecture [11] show that the standard REINFORCE learning rule used in their work and in that of Mnih et al. [10] is equivalent to approximately optimizing the free energy.

Van de Maele et al. [74] show that their unsupervised generative model exhibits information-seeking behaviour during next-best-view selection in a slightly trivial, simulated setting where a robot arm is tasked with finding a block of a specific colour. The generative part of their model is trained as an autoencoder that uses an in-hand camera's prior observations to predict observations from new viewpoints. The next view policy is selected by evaluating a number of candidate policies and selecting one with the lowest expected free energy, i.e. the lowest surprise. In a follow-up work, Van et al. [75] update their approach with a variational auto-encoder and apply it to a subset of ShapeNet [76] and two other virtual environments; however, the efficiency of their approach is still limited by the need to evaluate numerous candidate viewpoints.

### 3.2.6 Other Relevant Literature

In a paper titled "Active Vision in the Era of Convolutional Neural Networks" Gallos & Ferrie [77] argue that calibrating a model's outputs, i.e. ensuring that they estimate the model's confidence in its predictions, is necessary for temporal reasoning of active vision systems. They also argue that calibration promotes view invariance, which in turn better enables an active vision system to cope with unexpected and low-value observations. In their literature review they describe how prior to the deep learning boom active vision literature focused on utilizing uncertainty and errors in choosing the next best view, and how active vision in deep learning has departed from that approach in favour of having the agent learn where to look end-to-end using reinforcement learning. They incorporated uncertainty into their own model by using MC-dropout [78] with a dataset of objects viewed from multiple angles. Although their argument is principled and well-informed by past literature, their experimental results do not provide very strong evidence in support of it as they do not show significant improvements over their benchmarks.

Cheung et al. [79] use a recurrent neural network trained to perform a simple visual task that involves using a visual sensor to search for and classify an object in a noisy scene. The visual sensor is a learnable retinal sampling structure, meaning that the parameters of its receptive fields are adjusted during training. They find that this setting gives rise to a foveated sampling pattern, providing more support to the idea of using foveation within active vision in deep learning.

Lukanov et al. [80] describe an active vision system that utilizes both top-down and bottom-up attention to drive a foveated sensor around an image. They efficiently encode the visual data captured by their foveated sensor using Foveal Cartesian Geometry [81]. The limitations of their system are that it is memory-free, restricting its capacity to solve tasks other than image classification, and that it uses a simple mechanism for bottom-up attention which is subject to a significant assumption. In this mechanism the next fixation is located at the maximum of the channel-wise average of their CNN's output feature maps; the authors assume that high values in the feature maps correspond to salient areas of the scene. Their top-down attention mechanism utilizes class activation maps [82], but it is limited by requiring the user to select the target class for the model to search for.

In a paper that predates the current deep learning boom by more than two decades Schmidhuber & Huber [83] describe an architecture for aligning an artificial fovea with a target object. Aside from the artificial fovea, the architecture also consists of a controller network that controls the fovea and a model network that predicts the future foveal view when given the current view sampled by the fovea and its next action. The architecture is trained in two phases, with the first phase being unsupervised and focusing on training the model network using random fixations. In the second phase the difference between the model network's predicted view and the desired view is used to compute gradients to train the controller network. This way the authors bypass the need for reinforcement learning by finding an analogous substitute for backpropagating the gradients through the environment.

Monica & Aleotti [84] train a CNN for choosing the next best view in a 3D environment exploration task using a depth camera. They use an autoencoder to perform object completion and infer the probability of voxel occupancy in unobserved areas and to create a probabilistic map which is then used to guide the sensor.

The Saccader model introduced by Elsayed et al. [85] is a hard attention model that incorporates passive vision and is thus not a strict active vision model. It first selects multiple parts of the image and computes sets of 2D features at different locations in parallel, and then uses an attention model with a memory cell to select one of those locations as the center of the next attention patch.

Fu et al. [86] propose a model that learns to iteratively zoom in on valuable areas within its field of view. It is not an active vision problem but it is closely related to foveated active vision as focusing the fovea on a part of the image is similar to zooming in on it. Their model consists of an ensemble of networks that have a separate convolutional feature extractor and classifier for each zoom scale, and a recurrent attention module in between each two scales that is used to propose regions to zoom in on. Each attention module takes as input the encoded features from the coarser scale to choose an area within it for magnification that will yield the image at the next scale.

Kyrkou [87] describes a feedforward network trained end-to-end to provide pan and tilt camera control for target monitoring. His network does not utilize memory and utilizes bounding boxes as labels for training the network to bypass the need for reinforcement learning.

### 3.2.7 Discussion

As the citation metrics are not conveniently available to the reader, it is worth noting that although active vision models which do not aggregate information from multiple observations, i.e. do not maintain a memory, have been described in the literature, they have received little attention and have not been widely cited. The most cited and improved upon architectures utilize a recurrent network to maintain a memory state and are described in Section 3.2.2. Amongst all of the reviewed literature the recurrent models of visual attention represent the most coherent and mature research thread, with many different research teams contributing to it over the years. This is likely motivated by the fact that active vision systems without memory are severely restricted in terms of their functionality as they can do little more than select the next best view, which has limited applications and utility for complex vision tasks such supporting an autonomous agent in navigating a real-world environment.

Overall, the active vision in deep learning literature is difficult to browse. The research thread investigating recurrent models of visual attention stands out in part due to the lack of similarly coherent research threads in the field. Many publications suffer from a lack of adequate referencing, with one example being present in Jayaraman et al.'s work [13] [14]; the LookAhead module that is the core aspect of their contribution is not strictly novel as it is only performing a part of the function of the model network from Schmidhuber & Huber's work [83]. Jayaraman et al. do not cite the latter's work in either of their two publications utilizing this module. The terminology used by different researchers to describe their work is also inconsistent, making it less likely for those who are not yet familiar with active vision to find all of the related publications. This is further compounded by the fact that research pursuing attention in computer vision has a significant overlap with active vision and is ignorant of the terminology used in physiological literature describing visual attention in living beings.[2] As a result there is a reduced likelihood of computer vision researchers adopting active vision or any other biomimetic inspiration in their research, and interesting potential projects such as mimicking the active vision of bees [88] [89] [90] for developing autonomous aerial drones may never be pursued. The author of this thesis believes that

[2]As previously explained in Section 3.2.1, the term "attention" is being used in deep learning as a catch-all term that usually corresponds to the term "covert attention" in physiological literature. Occasionally, deep learning literature also refers to active vision as simply "visual attention".

a significant knowledge gap in the literature is the lack of a literature review aggregating different active vision publications and establishing standard terminologies and taxonomies for referring to different components of, problems in and approaches to active vision in deep learning. This review and the research framework described in Section 3.4 aim to address this knowledge gap.

When deciding on an approach for enabling their model to choose where to look, most researchers in the reviewed literature have either opted for reinforcement learning or utilized denser labels that include data on the salient locations in the training images. Both of these approaches are simple, if not the simplest, ways of solving the problem of learning an attention policy. Choosing the most simple solutions is an adequate research strategy when dealing with problems that do not have much prior literature describing how to tackle it, but as this literature review has shown, there is even an approach for training the attention component of an active vision system without reinforcement learning that dates back to the 90s [83]. Instead of contributing novel solutions or building upon previously introduced approaches to choosing where to look, most recent literature takes the safe approach and rehashes the same basic and simple ideas. There are many broader issues in the deep learning publishing ecosystem that one could bring up to explain this; however, the author of this thesis believes that this is because much of the relevant prior active vision literature is hard to come by for reasons already mentioned above.

This observation is similar to that of Gallos & Ferrie [77], who criticised the field for abandoning the use of principled scientific methods for guiding the agent in favour of end-to-end methods. Model-based approaches such as that of Schmidbuber & Huber [83] have not been expanded upon within active vision literature; however, the methods explored under the banners of active inference in Section 3.2.5 and embodied AI in Section 3.2.4 develop novel approaches that are arguably more principled than relying on reinforcement learning to solve everything. The author believes that a standard investigative method for evaluating how well an agent chooses where to look in a way that is not confounded by its feature extraction capabilities or sensor design would facilitate more research into novel algorithms for choosing where to look.

A potentially confusing feature that has been a part of the active vision literature since the term was first coined is that different researchers refer to different problems and aspects of active vision with the same term. Aloimonos et al. [4] use the term to refer to an active observer that samples images from different viewpoints, whereas much of the argumentation presented in Ballard's work [9] is agnostic of viewpoints and instead refers to the benefits conferred by an active observer locking their gaze on a moving object. The novelty in the active visual recognition method described by Cheng et al. [16] lies in the way they aggregate information about a 3D scene from multiple observations, which is an entirely different problem from next best view selection that makes up a significant proportion of active vision literature. There are many possible motivations and strategies for a visually endowed agent to actively explore their surroundings; consequently, different visual tasks can pose different requirements on active vision, which in turn can confer distinct benefits in each of those settings. This complexity is not reflected in the literature and as a result many of the published works pursue disjointed approaches that do not reinforce each other in spite of ultimately striving for the same set of behaviours in vision systems.

While at first glance it may seem to be a simple task, choosing where to look can mean solving radically different lower level vision problems in completely different settings and for different reasons. The objectives motivating active exploration vary, with most of the literature reviewed focusing on optimizing the sampled information for object or image recognition. In contrast the system described by Monica & Aleotti [84] aims to exhaustively explore a 3D space, whereas the work of Chaplot et al. [70] done under the umbrella of embodied AI focuses on an active exploration policy for training their agent.

A significant proportion of the active exploration in the human visual system exploits foveation and can therefore be understood as a form of predicting high-frequency information content of the blurry peripheries of our vision; this is in line with Findlay & Gilchrist's [53] characterisation of covert attention as a pre-processing step in active vision. In contrast, the active vision datasets introduced by Ammirato et al. [6] and Malmir et al. [63] [64] focus on sampling entirely new information and do not incorporate any form of foveation, thus framing the problem of choosing where to look as predicting the completely unseen space beyond the agent's field of view. The former case using foveation is related to interpolation and super-resolution algorithms, which recover high-resolution images from lower-resolution inputs; whereas the latter case is more related to extrapolation and image in-painting algorithms, which fill in unseen regions in input images. Both of these ways of understanding visual exploration fall under the umbrella of active vision, with each one being tailored to a different visual setting, yet they have not been sufficiently differentiated in the literature and they do not have standard means of being individually evaluated in deep learning systems.

In spite of often producing valuable insights, much of the work described in the literature either does not strictly follow active vision requirements by employing a passive vision component that does not have a limited field of view, or simplifies the task and the setting used to evaluate the system. The active vision systems described by Li et al. [12] and Sermanet et al. [15] use virtual sensors with very large fields of view, while the system proposed by Mathe et al. [67] utilizes perfect memory. Ba et al. [11] and Elsayed et al. [85] employ passive vision in their models and many other publications described in this review rely on so-called toy datasets. All of these aforementioned design choices, while often intentional or necessary, simplify active vision to the point of trivializing the associated problems instead of advancing our understanding of how to solve them. The author believes that many of these decisions are due to researchers prioritizing the pursuit of high benchmark scores as opposed to producing insights or enabling their systems to exhibit novel behaviours.

## 3.3 Pilot Experiments

### 3.3.1 Motivation and Objectives

The literary survey conducted above has uncovered several shortcomings in the literature, but there are insights that can only be gained by experimentally evaluating the ideas presented within the literature. A number of relevant questions have been omitted entirely in the literature, while some design choices have been presented without any supporting evidence. The main goal of the experiments described in this section is to test how well solutions employed in the literature hold up when deployed in a setting that is strictly tailored for evaluating active vision. Additionally, the experiments aim to find out whether several questions that were not explored in the literature matter, as well as to evaluate the software retina in a more feasible setting than the EPIC-KITCHENS dataset

The first research question posed in this section asks about the optimal coordinate frame for the active agent to operate in. Ballard [9] has argued for the benefits of using an exocentric coordinate frame in a way that raises the question of whether the coordinate frame used by the active vision agent has any influence on its performance, yet this line of thought has not been explored any further in deep learning literature. The specific setting and use case that Ballard was referring to cannot be reproduced with the chosen materials as Ballard's use of the term “exocentric coordinate frame“ referred to the active agent locking their vision onto a moving object. Instead, in this section an exocentric coordinate frame is defined as the absolute pixel coordinates in the input image, as the input image is considered to be a substitute for the environment that the active vision system operates in. In contrast, an egocentric coordinate frame is defined as coordinates relative to the agent's current fixation location and normalized by the sensor's width. This formulation makes the agent agnostic as to the global location of its fixation in the input image. The hypothesis associated with this research question is that, if given the opportunity, the agent will exploit the image's absolute coordinates in a way that facilitates overfitting and prevents it from learning a generalizable attention policy.

Mnih et al. [10] and Ba et al. [11] have provided their agents with proprioceptive data in the form of their current fixation location. Investigating the impact of this design choice is the next research objective in this section. Although the aforementioned researchers have not motivated using proprioception, hypothetically it could be advantageous by enabling the agent to encode past fixation locations in its memory and learn inhibition of return, i.e. learn to avoid re-visiting previously seen parts of the scene. Another hypothetical benefit of proprioception is that it could facilitate more intelligent reasoning about features that have been extracted from distant regions of the scene by exposing their relative locations.

Another research question asks about the impact of the agent's visual sensor, specifically its foveation and its field of view, on classification performance. Among other researchers Li et al. [12] have given their agent a non-foveated sensor with a very wide field of view, which hypothetically risks trivializing key problems associated with active vision. Maintaining a constant sensor field of view has not been argued for in active vision literature. The author believes this to be problematic, as the lack of this practice could complicate interpreting research results if active vision agents' performance is significantly affected by its field of view. One of the research objectives in this section is to evaluate the performance of various visual sensors, including the software retina.

An overwhelming majority of active vision research in deep learning utilizes reinforcement learning for training the attention network. The dataset used in this section enables ablating the attention network by guiding the model's gaze onto areas of the image that ought to be the most discriminative of the different image classes. This method will be used to evaluate the effectiveness of reinforcement learning in producing an attention policy, as well as to help clarify the relationships between the attention policy and the aspects of active vision that are the focus of other research objectives, such as the agent's coordinate frame.

### 3.3.2 Materials

All experiments in this section were conducted using Pytorch 1.10.0+cu102 on a PC running Ubuntu 18.04 with a GeForce GTX 1080 Ti and an Intel Core i7-7700.

#### Caltech-UCSD Birds 200-2011

The dataset used in this section is the Caltech-UCSD Birds 200-2011 dataset [61], from here on referred to as the Birds datset, and it has been chosen to succeed EPIC Kitchens as the dataset of choice for the remainder of this dissertation. It is a relatively small dataset that consists of 11788 photographs of 200 bird species, additionally labelled with anatomical part locations. It is categorized as a fine-grained image classification dataset, as differentiating between some of the bird species requires taking into account fine visual detail. This characteristic makes the dataset uniquely suitable for evaluating foveated active vision systems, as fixating on locations containing the key fine details should be a pre-requisite for correct classification. Compared to EPIC Kitchens the dataset is smaller and much simpler while still consisting of natural scenes, making it a much more appealing asset for this research. An example image with anatomical part location labels from the Birds dataset can be seen in Figure 3.1.

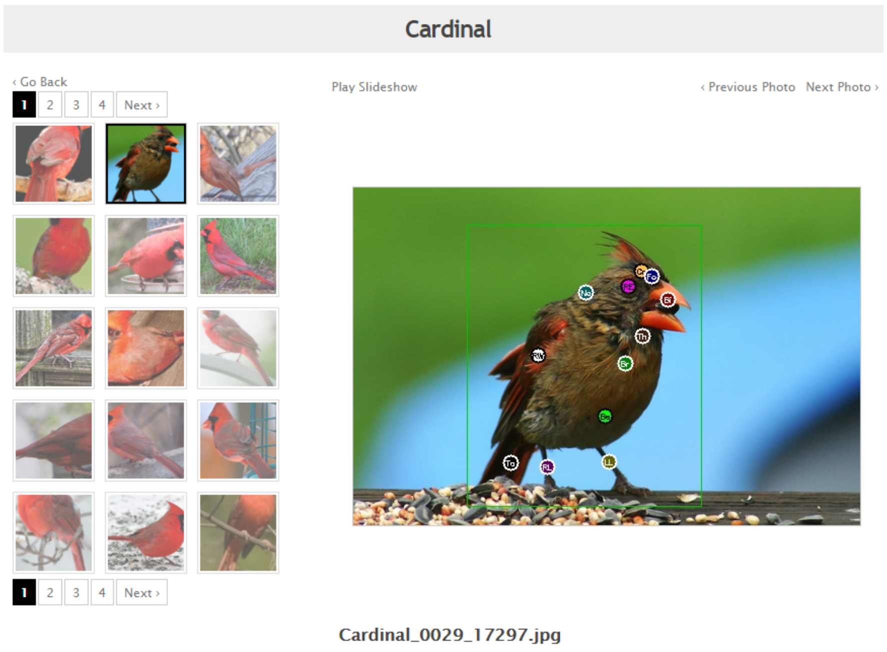


Figure 3.1: An example image, with part and bounding box labels, from the Caltech-UCSD Birds-200-2011 dataset, as seen on the dataset website.

Two other candidate datasets were considered for use in this project: the GERMS dataset introduced by Malmir et al. [63] [64] and the dataset for evaluating active vision introduced by Ammirato et al. [6], from here on referred to as the AV dataset. The Birds dataset only enables an exploration of a 2D image, whereas the other two datasets explore 3D scenes; the author believes that this is the only significant weakness of the Birds dataset. The Birds dataset provides a setting in which the action space is effectively unrestricted, meaning that an active vision agent can explore the image freely without any limits to or discretization of its fixations beyond the limits imposed by the image boundaries and the discretization imposed by the image pixels. In contrast, the GERMS and the AV datasets only allow the model to select from a small set of discrete actions which trivializes the problem of choosing where to look in a way that cannot be easily overcome with experimental design. The Birds dataset does not impose such a restriction and its user can choose whether and how to restrict the model's action space.

The discretization of the model's action space combined with the datasets forcing the model to explore the scene by selecting new camera views complicates the evaluation of agents with foveated sensors. To be able to fixate the high resolution fovea on any location in the scene the agent would have to choose at each timestep whether to select a new view using the physical sensor modelled by the dataset, or whether to move a virtual sensor within the bounds of the current view, as is being done in the active vision setting employed in this dissertation. Such a setup would introduce a redundancy that would not translate well to real-world problems as it would be challenging to justify simultaneously employing two forms of active sensors; a virtual one and a physical one. In the author's view virtual sensors ought to only be employed in active vision systems either in research settings to emulate a hardware sensor or in tandem with a high field of view camera that produces too much data to be adequately processed with a passive system.

The Birds dataset is more visually rich and diverse than the GERMS dataset, which was captured in very monotonous settings where the same robot arm in the same position is executing the same motions in the same lighting conditions across all samples. At the same time the Birds dataset is simpler than the AV dataset, which explores quite a large 3D scene spanning multiple rooms.

The final strength of the Birds dataset relative to the other two candidate datasets is the fact that it includes the birds' anatomical part locations, which can be used to guide an active vision agent to the key location in each image. This additional source of labels can be used for ablating the model's attention policy in order to evaluate its other components in isolation without resorting to deploying them in a passive setting.

### Modified Recurrent Attention Model

The active vision architecture selected for the experiments conducted in this section is based on the Recurrent Attention Model, abbreviated as RAM, that was first introduced in the work of Mnih et al. [10] and was expanded upon in the publications described in Section 3.2.2. Out of all the variants described in that section, the devised architecture is most closely related that of Li et al. [12]. The RAM architecture is the most studied and built upon design in the literature. It is also easily extensible and is not subject to any inherent restrictions or simplifications that would trivialize any aspect of active vision, making it the optimal choice for this project.

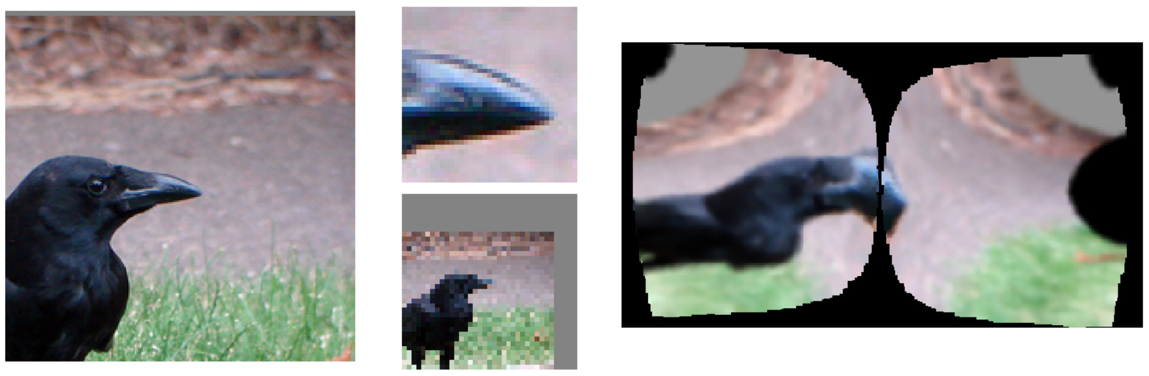


Figure 3.2: Example images produced by the 3 sensor variants used in this section. Not to scale. **Left:** the "large patch" sensor. **Middle:** the "foveated patches" sensor. **Right:** the software retina.

All images input into the architecture are subsampled by a virtual sensor with a restricted field of view. The experiments in this section utilize three variants of the sensor, with one of them being a software retina with 16,384 nodes, a field of view of 372*px* and a corresponding cortical mapping. The other two sensor variants extract square patches out of the input image; one variant crops out a full resolution patch that is 224*px* wide whereas the other one crops out two foveated patches, with the foveal patch being 37*px* wide and the peripheral patch being 370*px* wide downscaled by a factor of 10. Images produced by the three different sensors can be seen in Figure 3.2.

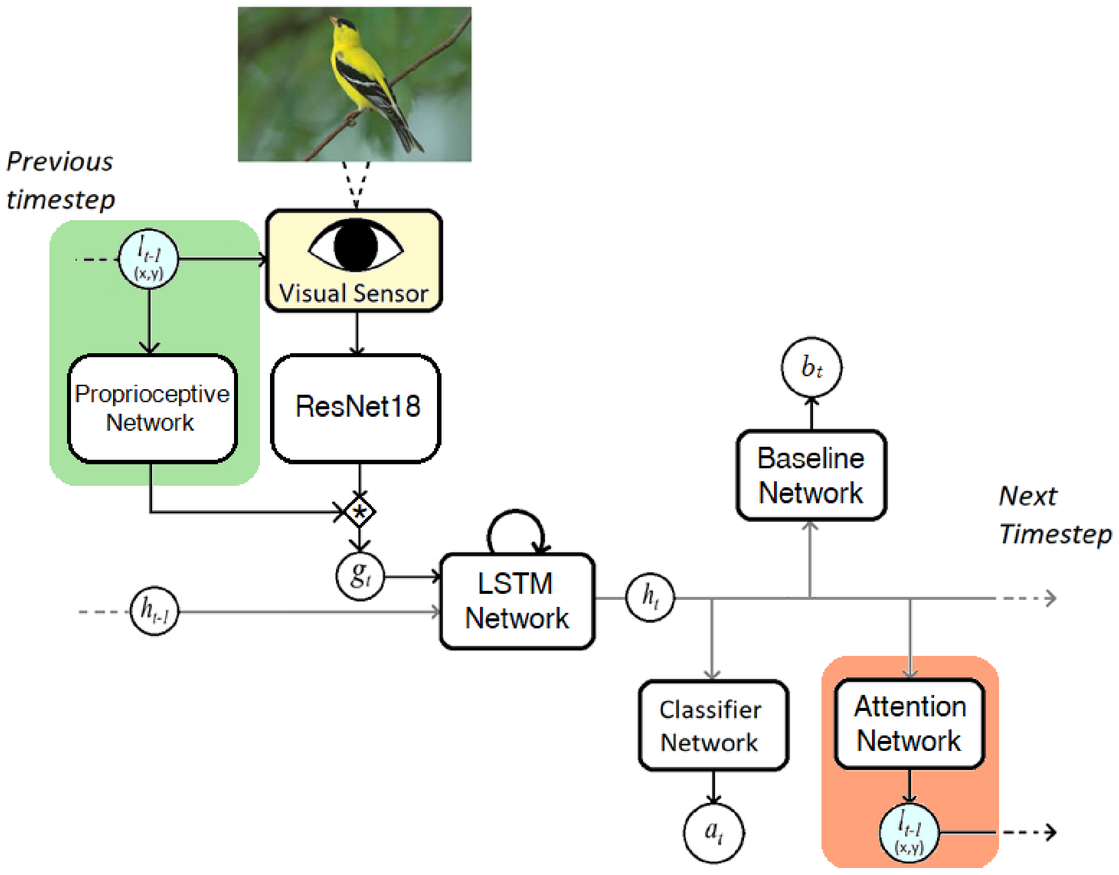


Figure 3.3: A generic diagram of the modified recurrent attention model used in this section. Colour coding corresponds to the colours used in Table 3.1. In the diagram *t* stands for the current timestep, *g* is the glimpse tensor representing all of the captured visual and proprioceptive features, *h* is the LSTM hidden state, *b* is the predicted reinforcement learning reward, *a* is a tensor of classification scores and *l* is the fixation location. Multiple variants of this architecture are trained, with the attention, baseline and proprioceptive networks being occasionally ablated. The different variants are listed in Table 3.1.

Figure 3.3 visualizes the architecture used in this experiment. The visual data captured by the sensor is passed to a modified ResNet18 feature extractor [47] which, in tandem with the proprioceptive network, act as an equivalent of the original RAM's glimpse network that was previously described in Section 2.4.2. The output of the proprioceptive network is a high dimensional embedding of the fixation coordinates that is combined with the features extracted by the ResNet18 to produce the glimpse vector $g_t$. The strides in the ResNet 18's *layer4.conv1* and *layer4.downsample* layers were set to 1 and its final fully connected layer was removed. In the case of the two patch sensor each patch is passed through ResNet18 separately and the two outputs are concatenated along the channel dimension. Depending on the experiment, the architecture may also utilize proprioceptive information in the form of the last action taken by the attention policy module encoded by the x,y coordinates of the current fixation point, denoted with $l_{t-1}$. This information is processed by the proprioceptive network, which consists of one fully connected layer with a ReLU activation that outputs a high dimensional embedding the dimensions of which match the dimensions of the ResNet18's output, which is (1024,2,2) for the foveated patches sensor, (512,7,7) for the singular image patch sensor and (512,9,12) for the software retina. These dimensions vary because the output side of the convolution operations inside ResNet18 depends on the input size, which in turn varies for each of the sensors used. The tensor produced by the proprioceptive network is combined with the visual features extracted by ResNet18 via multiplication to produce the glimpse vector $g_t$; combining the two data streams in this manner was done at a lower dimensionality in the DRAM paper [11], the authors of which claim to have taken the idea from Larochelle & Hinton [91]. The resulting tensor is adaptively average pooled, collapsing its spatial dimensions, before being passed to an LSTM network as $g_t$.

The LSTM's input size is equal to the channel dimension of the ResNet18's output, and it has 1024 hidden units making up its hidden state, denoted with $h_t$. At the first timestep its hidden and memory states are both initialized to zeros. During every timestep the attention network takes the LSTM's hidden state as input and returns the next fixation coordinates ($l_t$) together with two values for the backpropagation of reinforcement learning loss which are omitted from Figure 3.3 for simplicity. The attention network consists of two fully connected layers, with their input and output sizes being (1024, 512) and (512, 2) respectively. The first layer utilizes the ReLU activation function, and the output layer utilizes the the tanh activation function. At training time, the attention module comes coupled with a baseline network that serves to stabilize reinforcement learning. The baseline network is trained to predict the reinforcement learning reward that the attention network is going to get; this prediction ($b_t$) is subtracted from the actual reward in order to reduce the variance of the attention network's loss.

Full details about the attention module's training algorithm used in this section can be found in the original RAM paper [10] as well as the DRAM paper [11]. Some experiments do not utilize the attention module and instead use the anatomical part locations provided in the Birds dataset. At the final timestep the LSTM's hidden state is passed on to the classifier module which consists of a single fully connected layer with the log softmax activation function that maps the 1024 unit LSTM hidden state to 200 classification scores for the current image. In this section the model is set to execute for 3 timesteps per image. Exhaustive architectural optimisation was not performed for the architecture employed in this section as maximizing classification accuracy on the Birds dataset was not necessary for answering the research questions.

### 3.3.3 Methodology

As previously mentioned in Section 3.3.1, the research questions that the experiments in this section aim to answer are:

1. What is the optimal coordinate frame for the active vision agent to operate in?
2. Does exposing the active vision agent to proprioceptive data in the form of the current fixation coordinates improve its performance?
3. How will the active vision system be impacted by the use of a foveated sensor when compared to an unfoveated high resolution visual sensor?
4. How does reinforcement learning compare to using hardcoded locations in fixating the retina on valuable locations?

A total of 9 variants of the generic architecture shown in Figure 3.3 were trained. The specifics of their architectural aspects were selected so as to enable achieving the research objectives described in Section 3.3.1; they can be found in Table 3.1. The variant numbers in that table correspond to the variant numbers used in figures in Section 3.3.4. Each variant has been trained with three different random number generator (RNG) seeds - 1, 9, 919 - for network weight initialization, except the software retina variant (#7) which was trained with only one seed due to time constraints. Variants utilizing hardcoded fixation coordinates and without proprioception (#5, #6, #7) do not take as input or output their fixation coordinates, making the coordinate frame aspect not applicable to them.

To answer the research questions the relevant architectural variants will be compared against each other by examining the loss and accuracy plots produced during training. Analyzing plots is preferable to solely reporting the best results achieved during validation as interpreting them is less vulnerable to outliers and they can provide more insight into the model's behaviour. The retina variant will also be compared against the others in terms of real time taken for the completion of one epoch.

Each variant was trained using the same algorithm. The training and validation splits were provided as part of the Birds dataset. The ResNet18 feature extractor was initialized with weights resulting from training it on the ImageNet dataset [30]. The maximum training time was set to 100 epochs; however, early stopping was triggered whenever validation accuracy did not improve by at least 0.5% over the duration of 40 epochs. The batch size was set to

| | Architectural Aspects | | | |
|---|---|---|---|---|
| | Egocentric | Proprioception | Hardcoded Fixations | Sensor |
| #1 | ✗ | ✓ | ✗ | Foveated Patches |
| #2 | ✗ | ✓ | ✓ | Foveated Patches |
| #3 | ✓ | ✗ | ✗ | Large Patch |
| #4 | ✓ | ✗ | ✗ | Foveated Patches |
| #5 | N/A | ✗ | ✓ | Large Patch |
| #6 | N/A | ✗ | ✓ | Foveated Patches |
| #7 | N/A | ✗ | ✓ | Software Retina |
| #8 | ✓ | ✓ | ✗ | Foveated Patches |
| #9 | ✓ | ✓ | ✓ | Foveated Patches |

Table 3.1: The architectural variants trained in this section. Colour coding corresponds with the colours in Figure 3.3. Variants without egocentric coordinate frames use an exocentric coordinate frame. Variants without hardcoded fixations use the attention network trained with reinforcement learning. Only these 9 variants were required to answer the research questions; other combinations of the four architectural aspects would not contribute to the research objectives posed in this section.

16. The initial learning rate was set to $5.0e^{-3}$ and a scheduler was employed that cut the learning rate in half every 10 epochs. The optimizer used was stochastic gradient descent (SGD) with momentum set to 0.9 and weight decay set to $5.0e^{-3}$. There are three sources of gradients during training: classification gradients that originate at the classifier network and are backpropagated all the way to the input layer, reinforcement learning gradients that are backpropagated only through the attention network, and baseline gradients that are back-propagated only through the baseline network. The classification loss is computed using negative log likelihood. Full details about the attention network's training algorithm can be found in the original RAM paper [10] as well as the DRAM paper [11].

All variants are set to execute three timesteps before classifying the input image. Variants using the attention network will start at a random location in the image sampled from a uniform distribution. Variants with ablated attention policy utilize hardcoded fixations to fixate on three anatomical part locations: the birds' eyes, the birds' beaks and the birds' tails. During training the order in which these locations are visited is randomized; however, during validation it is not. Some samples from the Birds dataset are subject to occlusions, and the occluded body parts are not labelled with their locations. In those cases these missing locations are substituted for the locations of the approximately closest visible anatomical part.

All hardcoded fixation locations have normal noise with $\sigma = 4$ applied to them; the locations are then rounded to the nearest integer with a "round half to even" caveat. In order to exert finer control over the training process two separate RNG states are maintained: one for applying noise to hardcoded fixation locations and shuffling their order during training, and one for selecting samples from the dataset. Both of those RNG states are seeded with the value 303. The training and validation splits used were the ones provided as part of the Birds dataset.

### 3.3.4 Results

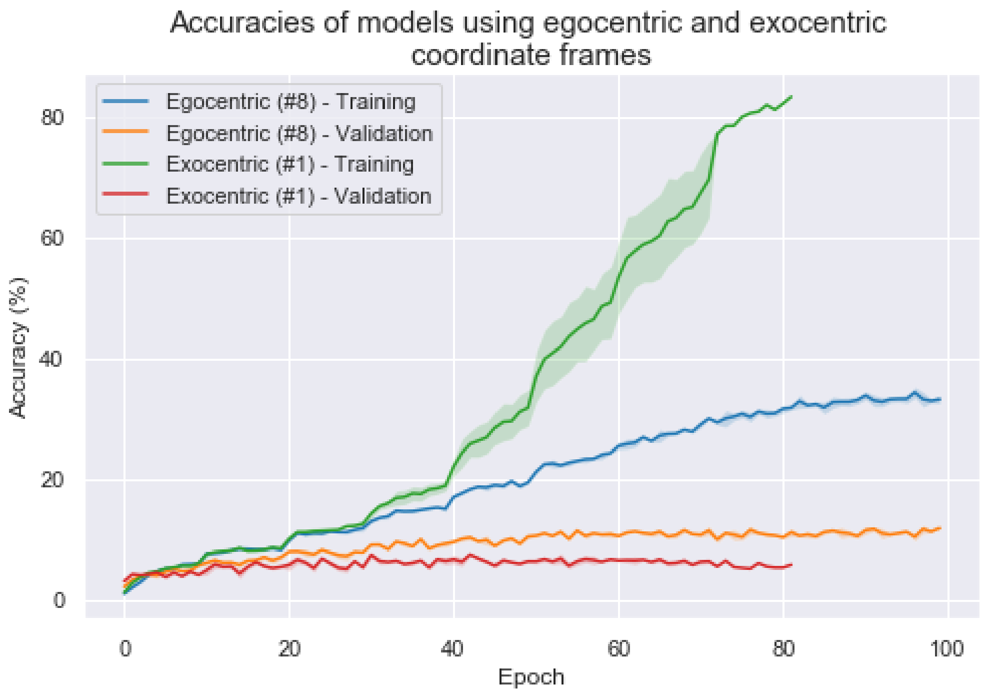


Figure 3.4: Accuracies of recurrent attention models using different coordinate frames. Lines represent mean values, shaded regions represent the standard deviation.

Figures 3.4 and 3.5 show how using an egocentric coordinate frame enables the model to outperform a variant using an exocentric coordinate frame. Due to the sensitivity of active vision to the starting conditions in image exploration, standard deviation has been calculated only with respect to different network initialisation seeds. For this reason the shaded regions representing the standard deviation are often narrower than expected in the loss and accuracy plots in this and following sections. The exocentric model can be seen to overfit the training set drastically; its classification accuracy at training time is much higher than that of the egocentric model, but its validation accuracy is significantly lower. Their training losses are not as divergent as their training accuracies; however, the validation loss of the exocentric model can be seen to grow drastically. These results support the hypothesis put forward in Section 3.3.1 which stated that exposing the model to absolute image coordinates will let it exploit those coordinates to avoid learning a generalizable attention policy. The two variants compared here are exposed to the coordinate frame via both proprioception and the attention network, which makes discovering which interface is responsible for these results difficult.

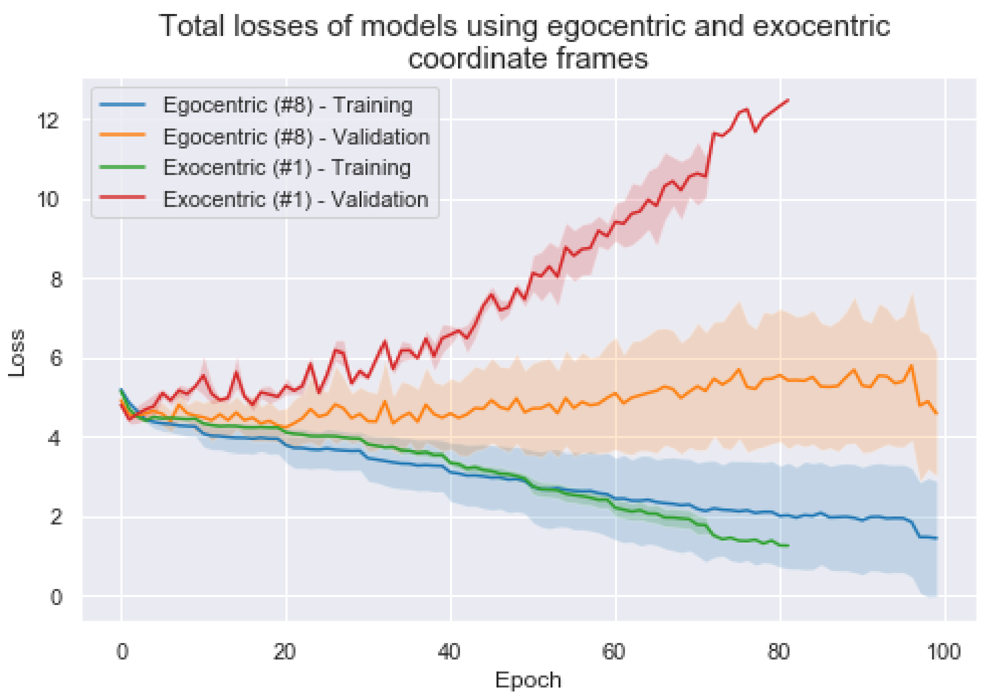


Figure 3.5: Losses of recurrent attention models using different coordinate frames. Lines represent mean values, shaded regions represent the standard deviation.

It is worth noting that the loss plots of the egocentric coordinate frame are noisy. Further results shown in Figures 3.8 and 3.9 suggest that this is caused by egocentric proprioception, as ablating proprioception causes the effect to disappear. This is most likely due to the fact that in these experiments classification gradients can travel into the attention from the proprioception module through the fixation location. These gradients could be a source of noise applied to the attention network weights; however, it is not clear why does the exocentric coordinate frame not produce the same effect.

Figures 3.6 and 3.7 show how the training behaviour of the models changes when their attention is ablated and replaced with anatomical part locations, as well as how a model with no proprioception behaves in the same conditions. This helps clarify the effect of the coordinate frame; the two frames result in effectively identical training behaviours which suggests that the impact of the coordinate frame is contained to the attention module and that the format of the data input into proprioception does not make a difference. Furthermore, removing proprioception shows improved performance on both the training and validation sets suggesting that proprioception is detrimental to the effectiveness of feature extraction, likely by introducing noise into the architecture's forward pass.

There is a significant gap in performance between models with ablated attention shown in Figures 3.6 and 3.7 and the models using an attention network shown in Figures 3.4 and 3.5. All models using anatomical part locations approach 100% accuracy on the training set, which is expected given the consistency of the visual input produced by each sample across

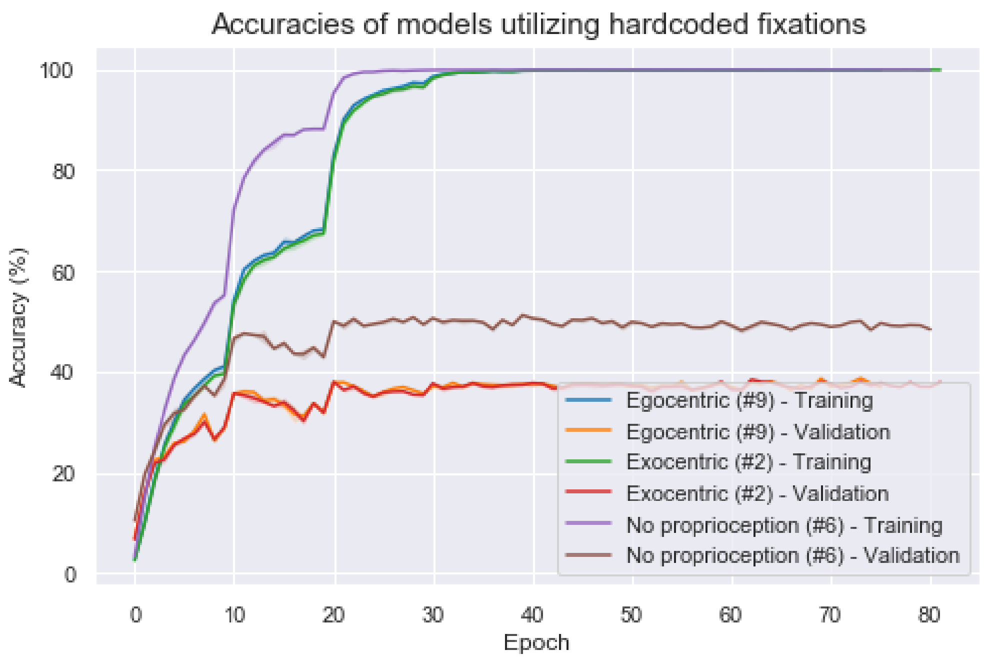


Figure 3.6: Accuracies of models with ablated attention using different coordinate frames. Lines represent mean values, shaded regions represent the standard deviation.

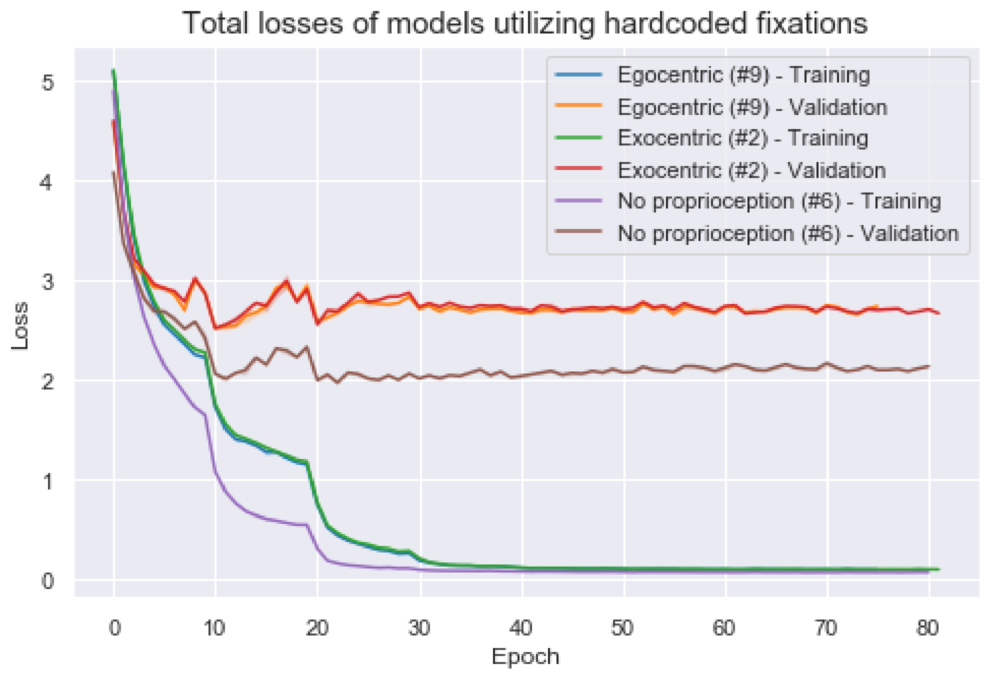


Figure 3.7: Losses of models with ablated attention using different coordinate frames. Lines represent mean values, shaded regions represent the standard deviation.

epochs. The model variants utilizing the attention network perform significantly worse on the validation set, in some cases even struggling to improve their performance relative to their initialization. This demonstrates that reinforcement learning is not adequate for training an attention policy for non-trivial active vision problems.

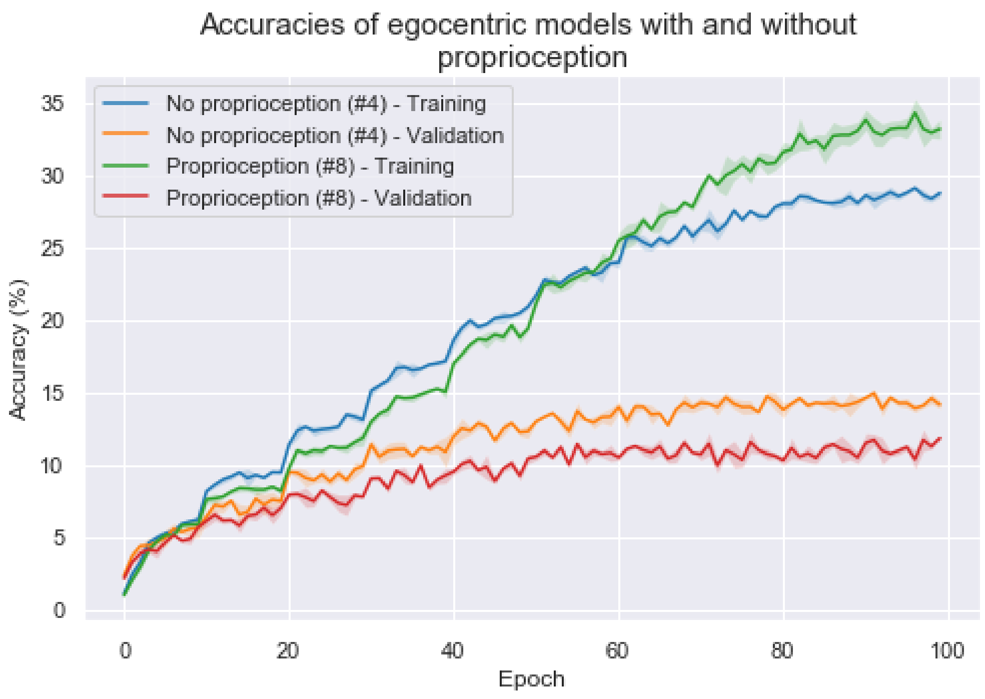


Figure 3.8: Accuracies of egocentric models with and without proprioception. Lines represent mean values, shaded regions represent the standard deviation.

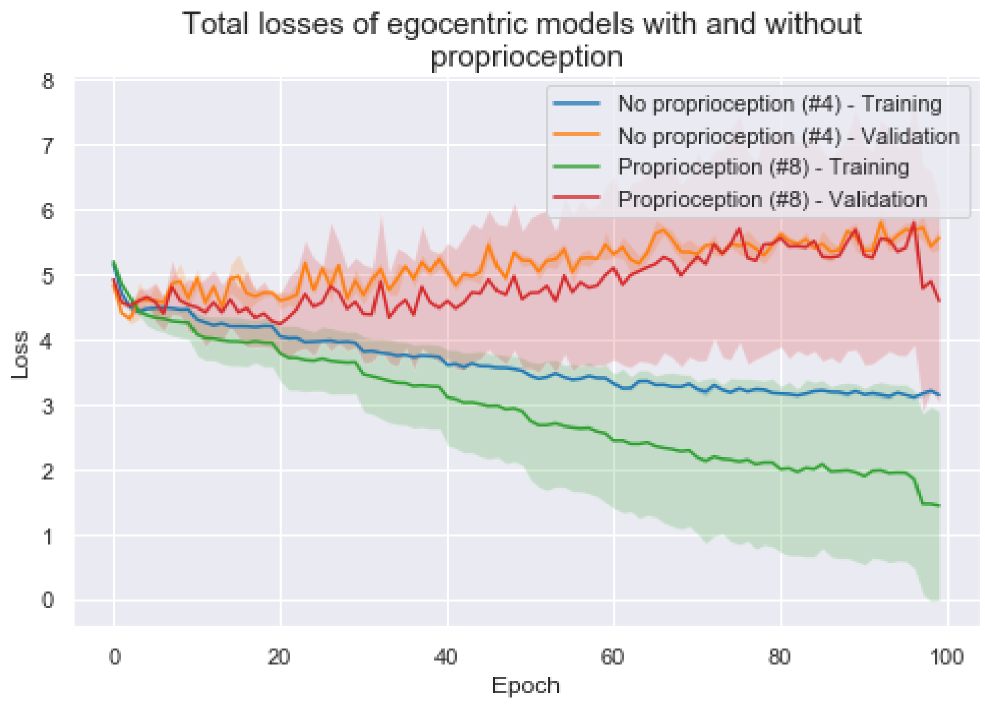


Figure 3.9: Losses of egocentric models with and without proprioception. Lines represent mean values, shaded regions represent the standard deviation.

The result of enabling the attention network with an egocentric frame and evaluating the impact of proprioception can be seen in Figures 3.8 and 3.9. A combination of the egocentric frame, proprioception and the attention network appears to be the source of high variance in the model's losses across different seeds; however, this could be due to re-running the experiments with an insufficient number of seeds as the seeds used could have simply been unfortunate for this variant. The accuracies of the proprioceptive model are not exhibiting as much variance as its losses. This could be due to the model's poor calibration, i.e. an overconfidence in mistaken predictions and an underconfidence in correct predictions, as it can affect the classification loss without a significant impact on classification accuracy.

Figure 3.8 shows that model variants without proprioception perform better on the validation set but worse on the training set than model variants with proprioception. This does not occur when the attention network is ablated as seen in Figure 3.6, suggesting that the proprioceptive input is causing an overfitting of the attention network to the training set

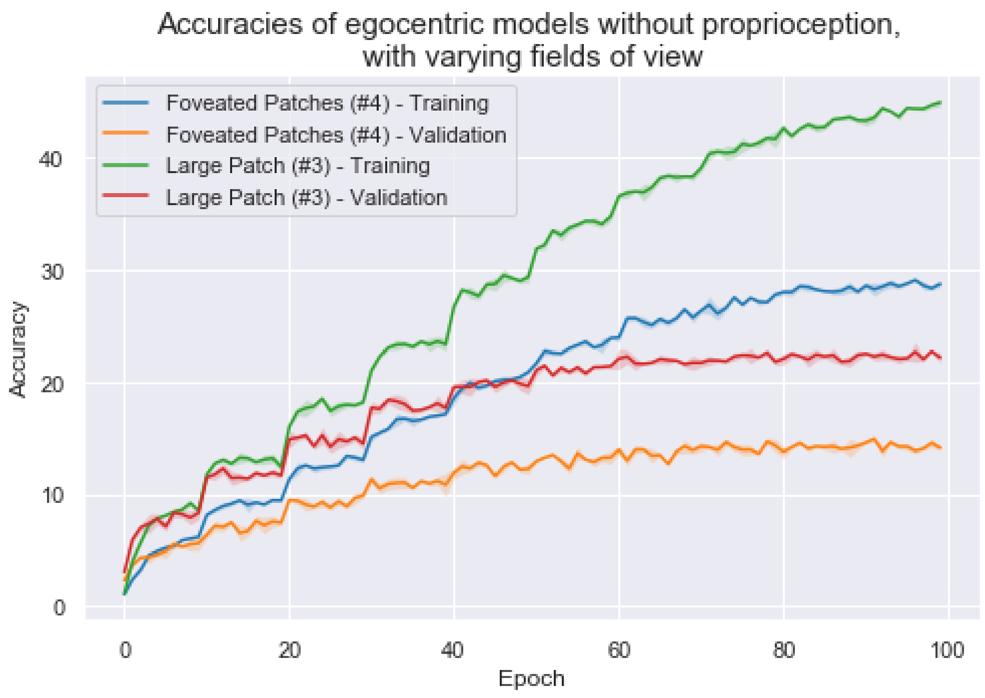


Figure 3.10: Accuracies of proprioception-free models with and without foveation. Lines represent mean values, shaded regions represent the standard deviation.

Figures 3.10 and 3.11 show the benefits of utilizing a sensor with a large full resolution field of view alongside the attention network. Even though the validation losses are similar, the validation accuracies of the two variants differ significantly, which could be due to a poor calibration of the unfoveated variant relative to the foveated variant.

The performance of model variants using various visual sensors with ablated attention and ablated proprioception can be seen in Figures 3.12 and 3.13. The software retina variant has the worst performance out of all the variants plotted in these figures and it took the longest to train: on average one of its epochs took 567.2 seconds to complete, whereas the non-retina variants took approximately 75 seconds per epoch.

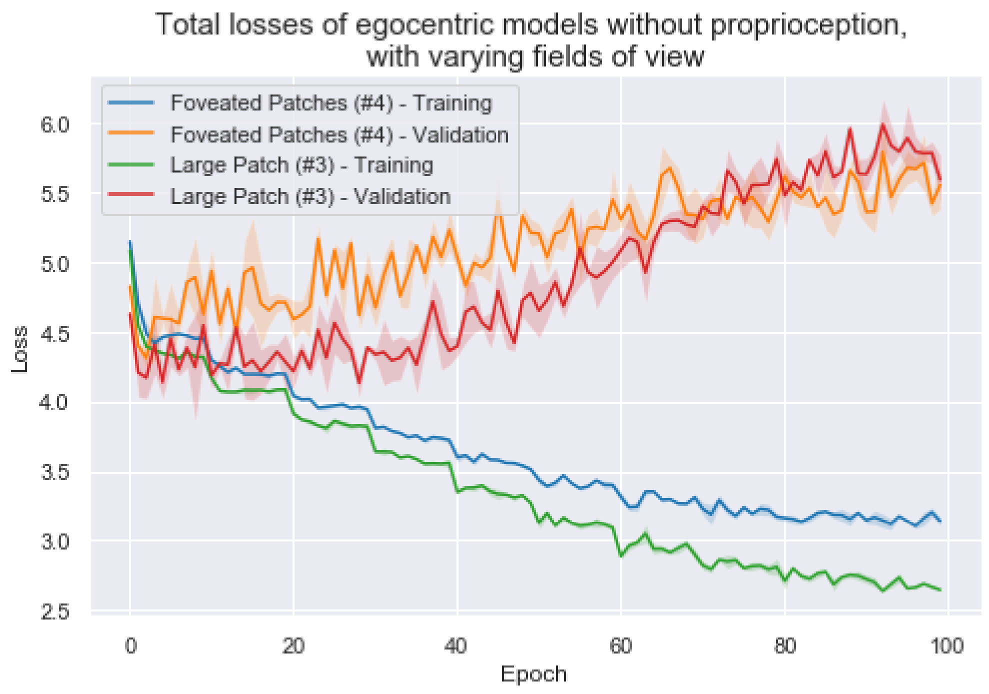


Figure 3.11: Losses of proprioception-free models with and without foveation. Lines represent mean values, shaded regions represent the standard deviation.

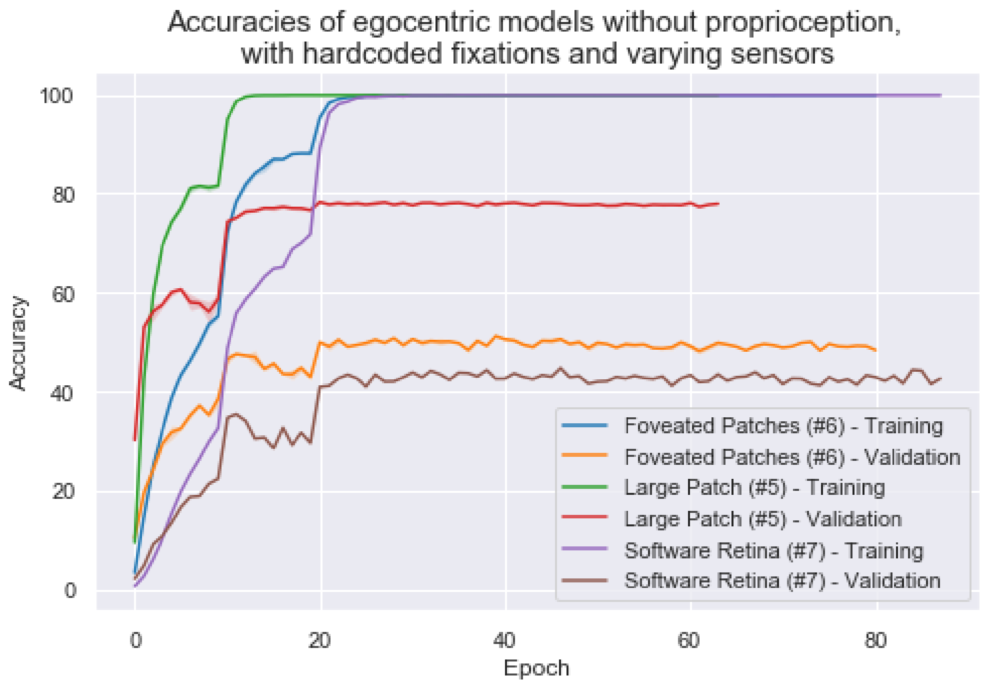


Figure 3.12: Accuracies of models using various visual sensors with ablated attention and ablated proprioception. Lines represent mean values, shaded regions represent the standard deviation.

With hardcoded fixations the losses of models with a large patch and foveated patches are no longer similar to each other, with the large patch model having half the validation loss of

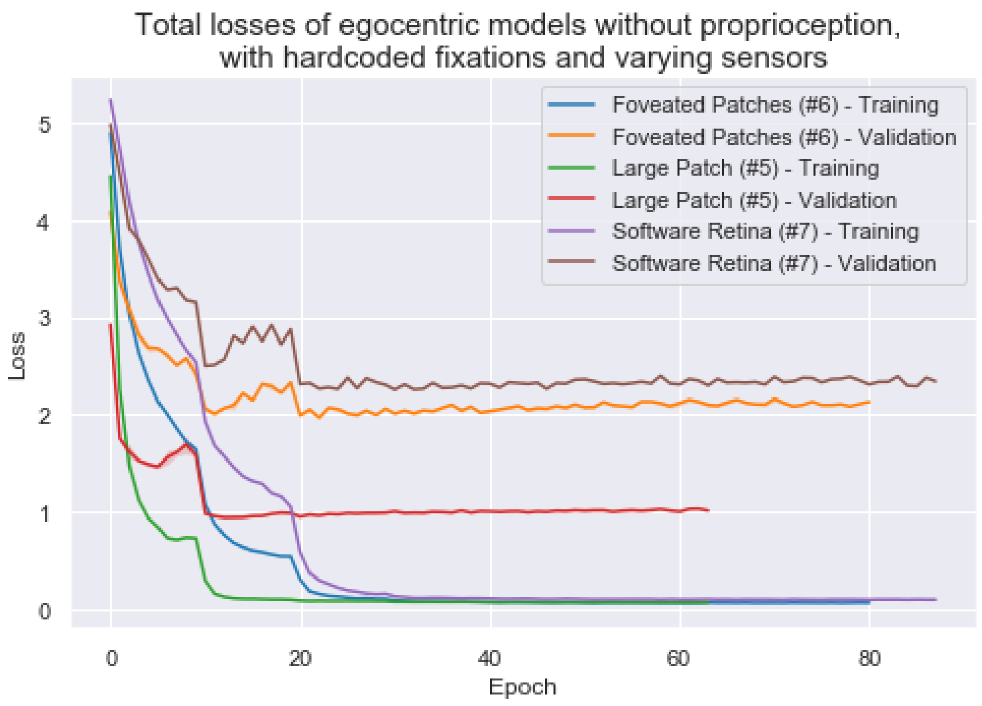


Figure 3.13: Losses of models using various visual sensors with ablated attention and ablated proprioception. Lines represent mean values, shaded regions represent the standard deviation.

the foveated patch model. The large patch model is almost 80% accurate on the validation set, with the foveated patch model being around 50% accurate. This large gap shows that there is significant work to be done on enabling the active vision architecture to effectively utilize information from multiple observations in a way that is competitive with approaches that utilize a large field of view.

### 3.3.5 Discussion

The experiments conducted in this section have provided insights about the state of active vision in deep learning that go beyond what was uncovered in the literature review. The results obtained regarding the sensor architecture, coordinate frames, proprioception and the effectiveness of reinforcement learning are critical of the literature by demonstrating that prior work has skipped past the fundamental groundwork that should have already been conducted. Instead of addressing these issues and investigating individual aspects of active vision, the literature focused on publishing papers about complete architectures that were trained in an end-to-end fashion.

One observation that was made in the literature review focused on the tendency of researchers to use simplified settings and solutions when evaluating active vision systems. The popularity of reinforcement learning and the use of sensors with a wide field of view are two examples of this tendency; the results presented in this section suggest that the wide field of view might have been used as a band-aid for masking the inadequacy of reinforcement learning for the task of producing a suitable attention policy. In the experiments, variants using reinforcement learning instead of hardcoded fixations (#1, #3, #4 and #8) have consistently performed poorly, in some cases even failing to reduce validation loss beyond the starting value in spite of becoming more accurate. The large patch sensor has greatly improved model performance every time it was employed, but a foveated model guided to the anatomical part locations performed better than a model that used the large patch sensor and the attention network, showing that increasing sensor field of view is not enough to fully compensate for the lack of a good attention policy. With an adequate attention policy the performance gap between the large patch and the foveated patch sensors is hypothesized to decrease as the model is allowed to make more fixations and collect more information about the scene.

Investigating the impact of different coordinate frames came about as a result of reviewing the work of Ballard [9] and realizing that absolute image coordinates may be abused by an active agent in one of several ways. One way would be for it to learn human biases that impact how are objects of interest framed in common computer vision datasets, such as the rule of thirds [5]. Another way would be to correlate exocentric location data with training images; the model could learn that when exposed to a certain background scenery it should fixate on specific absolute locations that do not necessarily fall on the bird in order to best discriminate the image class. In the experiments exocentric model variants have drastically overfitted the training set at the expense of their validation performance, supporting the latter hypothesis. This effect was found to be caused by the attention network and not the proprioceptive input, as ablating attention has caused models using the two coordinate frames to perform equally well. This result gains additional weight due to the fact that the vast majority of the reviewed literature has utilized exocentric coordinate frames.

Contrary to what was implicitly suggested in the literature proprioception was found to be detrimental to classification performance, as shown in Figures 3.6 and 3.7. However, Figure 3.8 shows that proprioception has had a positive effect on training set accuracy when the attention network was not ablated. It may be that proprioceptive input can be useful to learning an effective attention policy; however, it appears that the learned policy does not generalize to the validation set in the experimental setup used in this section. These results suggest that a hypothetical way to preserve the potential positive effect of proprioception on attention while suppressing detrimental effect on classification is to feed proprioceptive data directly into the attention network instead of multiplying it with visual features and routing it through the LSTM network; if true this would call into question the design decisions made in prior literature that integrated proprioception with vision.

## 3.4 The Active Vision Framework

The proposal of a framework for understanding and researching active vision in deep learning is motivated by the numerous issues with active vision literature that were uncovered during the experiments conducted in Section 3.3 and the literature review in Section 3.2. These issues can be reduced into two underlying problems that the proposed framework seeks to address: an incoherence across the field and a large number of research gaps pertaining to key issues.

A major cause driving the fields' lack of coherence is the frequent use of inconsistent terminology between different publications. Sometimes different problems end up being tackled under the same label, whereas on other occasions different terms are being used to refer to the same problem or solution. As a result, in spite of the fact that they all ultimately strive for enabling the same behaviours in vision, very few active vision research papers build on top of each other's achievements.

The experiments in this section have identified several gaps in the literature that have a high potential for confounding active vision research. The literature did not account for the impact of different coordinate frames and sensor fields of view, and only a few publications were found that attempted to meaningfully improve on reinforcement learning for obtaining an attention policy without resorting to passive vision. However, the author of this thesis believes that the most pressing gap in the literature is the lack of promising attempts at introducing mature, standard research practices that can help researchers produce insightful results. The two datasets that were introduced specifically for active vision [63] [6] contribute little besides simulating a setting that simulates visual exploration. There are no common methods for a more granular evaluation of the different aspects of active vision, and as a result most literature proposes and evaluates complete architectures without adequate ablations. Such contributions do not produce much, if any, transferable knowledge and are hard to interpret due to numerous confounding factors, further compounding on the fields' incoherence and disjointedness.

In a bid to improve the coherence of the field and help identify the gaps within it the proposed framework introduces a structured way of reasoning about the different aspects of active vision. It starts with the observation that in deep learning the vast majority of passive vision systems can be reduced to consist of two main architectural elements: a “backbone“ feature extractor for producing a useful representation of the input data and a mechanism for processing the extracted representation in a manner that is specific to the visual task at hand. Some common “backbone“ feature extractors include the ResNet architectures [47], GoogleNet [58] and ResNext [92]. A common mechanism for processing visual features in image classification is a simple fully connected layer, but more complicated tasks such as image segmentation tend to require more sophisticated solutions such as the architecture of

Mask-RCNN [93].

Under the proposed framework active vision complements the two elements of passive vision systems with three additional architectural and functional requirements. These are an attention mechanism, a sampling structure and a memory mechanism. The solutions to each one of those requirements can be organized into further sub-categories either by the approach they adopt or by the specific visual task that they are tailored for. The author believes that active vision literature should be structured in a way that reflects this organisation and that this could be achieved if publications maintained a narrow scope by explicitly focusing on individual aspects of active vision while being more clear about the context that their research exists in.

**The sampling structure** is the visual sensor that defines the active agent's local sampling strategy. Visual sensors can be either uniform or space variant, with foveation being the most common form of a space variant sensor. The software retina described in Chapter 2 is a relatively complex space-variant sampling structure, but there is very little research that produced useful knowledge for developing visual sensors in active vision and no research that tackled the subject directly within deep learning. Due to their evolutionary adaptations different animals often have different arrangements of photoreceptors in their eyes, such as the cat which does not have a fovea and instead has a slit-shaped region of higher photoreceptor density [94]. This suggests that one possible avenue for future sensor-related research is the pursuit of task-specific space-variant sensors, for example by learning them using methods such as the one described by Cheung et al. [79].

**The attention mechanism** implements an active agent's global sampling strategy and is the means by which it chooses where to look. Solutions to attention can either be closely integrated with the agent's visual processing machinery or operate independently of it. Independent approaches often require the user to implement solutions to task and domain specific problems that integrated neural networks can implicitly learn to solve. Two examples of such problems are inhibition of return, which is a means of ensuring that the agent does not revisit previously observed parts of the scene, and optical flow compensation, which involves differentiating between agent-generated and exogenous optical flow in the visual field. There are numerous objectives that attention in active vision can pursue and many of them impose their own unique requirements. Some examples of vision tasks that require different types of global sampling strategies are exhaustive exploration, recognition, target tracking and sampling information for training another part of the architecture as was done in the publications described in Section 3.2.4.

**The memory mechanism** implements an active agent's strategy for aggregating visual data collected across multiple observations. Memory can also be understood as a form spatio-temporal attention. Memory-free active vision is possible; however, it represents a trivialized approach that is by definition extremely limited in its applications. The author of this thesis believes that memory-free active vision is not worth investigating in this thesis since a major motivation for the pursuit of active vision is enabling vision systems to advance the state of active vision so as to solve less constrained problems in real-world-like settings. The limited field of view of active vision sensors means that occasionally an object of interest will not be observed in its entirety, and one function of memory is to 'stitch' the whole object out of fragmentary observations. An active agent can aggregate visual information at any point in the architecture; it can do so in feature space immediately following the feature extractor, at the network's outputs by, for example, averaging its predictions, within the “backbone“ feature extractor or exogenously as in the NARX architecture [95][3].

There are many potential ways in which the three aspects of the proposed framework can influence each other. It is easy to deduce the hypothesis that when closely integrated attention and memory mechanisms become mutually dependent; however, in some cases they may also develop dependencies on the visual sensor used in their training. The experiment results described in Section 3.3.4 demonstrated that a large sensor field of view trivializes the problem of attention. Foveated sensors imply that the attention mechanism's role is to predict high-frequency information content at the sensor's blurry peripheries in order to decide which area of the field of view is worth exploring with the fovea, whereas uniform sampling structures imply exploration beyond the field of view. The former is related to the interpolation and super-resolution algorithms, whereas the latter is related to extrapolation and in-painting algorithms. An implementation of memory with a mechanism for suppressing inputs produced by low value observations can make the architecture more robust to an otherwise inadequate attention mechanism; however, it is not clear whether such a mechanism needs to be designed explicitly or whether its function emerges implicitly when using the methods that are currently employed for active vision memory. Furthermore, in some cases the presence of the memory mechanism may motivate including feature processing independent of the 'backbone' feature extractor, such as the decoder-encoder network that processes the aggregated information in the work of Cheng et al. [16].

The close relationships between the three elements of the adopted framework have implications for active vision research. The efficacy of an attention mechanism can only be measured with a memory mechanism already in place. If the representation produced by memory acts as a bottleneck that imposes an upper bound on the agent's task performance then producing better fixations is not going to be reflected in commonly used performance metrics. This concern is a strong argument for investigating solutions to active vision memory first before moving on to the attention mechanism; however, the agent needs to fixate on different parts of the scene in order to enable an evaluation of memory. A similar relationship exists between the visual sensor and the other two elements. This issue demonstrates the urgent need

[3]The NARX architecture has not been applied to active vision in deep learning.

for evaluating each of the three elements of active vision in isolation. As a result being able to ablate different parts of the system by replacing them with well-understood and simple yet moderately performant solutions becomes a priority in active vision research. Attention for recognition can be completely ablated if the most informative locations in each image are known; although such data was present in the Birds dataset it is hard to come by similarly well-labelled datasets. Memory is relatively simple to ablate, as it can be replaced with approaches such as computing the mean of observations in feature space. The visual sensors used in active vision research are too simple to warrant any ablation beyond resorting to a uniform image patch in the cases when foveated patches are suspected to confound experimental results.

## 3.5 Conclusion

This chapter presents the work that grounded the project in a new direction after transitioning away from the software retina and onto active vision. The literature review in Section 3.2 has uncovered multiple issues with active vision literature, some of which are a lack of coherent and mature research threads, an overly holistic approach to architectural development and inconsistent terminology.

Further problems were identified during the experiments described in Section 3.3 where a number of questions ignored by the literature were investigated and found to be pressing while some assumptions made in it were called into question. The importance of the coordinate frame and the sensor field of view was highlighted, while reinforcement learning and proprioception were shown to be ineffective.

All of those critical insights into the state of the literature have informed the proposal of a framework for understanding and researching active vision that is described in Section 3.4. The framework reduces passive vision architectures into two elements and appends three additional components for active vision systems; it then discusses the interrelations between them and explains the implications for active vision research. An argument is made for the importance of granular, tightly scoped research and for using datasets that can support ablating attention by including annotations for the most informative locations in the images.

The following two chapters of this dissertation follow the structure of the proposed framework and explore active vision memory and active vision attention mechanisms. Visual sensors for active visions are not explored beyond this point in the dissertation as Chapter 2 has already covered the subject in sufficient detail by focusing on the software retina.

# Chapter 4

# Memory

***Abstract:*** *The research framework proposed in Section 3.4 argues that investigating memory should take precedence during early-stage active vision research. This chapter pursues the investigation of memory while following the previous chapter's prescriptions. Section 4.2 argues that investigating active vision memory should consist of hardcoding, appraising and sequencing single-use attention policies, whereas Section 4.3 implements those steps for the Birds dataset. The sequenced attention policies are collectively appraised using different methods in Section 4.4, revealing that a simple concatenation performs well as an aggregation strategy. Section 4.5 introduces and investigates the concept of temporal robustness, while Section 4.6 evaluates a number of different recurrent memory variants and finds that they fall short of simple concatenation, calling into question a fundamental assumption structuring much of active vision research in deep learning.*

## 4.1 Motivation and Objectives

The main goal of the project is to advance the understanding of active vision through developing and investigating a deep learning active vision architecture. This chapter supports that goal by investigating the issue of memory for active vision in accordance with the framework proposed in the previous chapter.

One of the prescriptions of the proposed framework is to conduct tightly scoped and granular research that focuses on individual aspects of active vision, i.e. sensors, attention and memory, as opposed to focusing on the development and description of complete architectures in an end-to-end fashion. This recommendation is motivated by the fact that previous research describing complete architectures struggle to produce transferable knowledge as their results are produced using specific combinations of datasets, training methods, constraints, problem definitions and architectural solutions that are rarely reproduced or investigated individually.

This chapter aims to produce transferable knowledge about active vision by restricting its scope solely to active vision memory. It explores different approaches to memory, including different aggregation strategies and training methods. In order to further improve the quality of knowledge produced, this chapter also demonstrates an investigative method for studying active vision that improves the interpretability of memory's function.

## 4.2 Controlling Observation Value

The experiments conducted in Section 3.3 have shown that reinforcement learning does not enable the active vision system to produce high quality fixation locations for an image classification task. As high quality observations are necessary to test whether memory is acting as a bottleneck on model's performance, the model's attention mechanism has to be ablated and replaced with keypoint locations, which in the case of the Birds dataset are anatomical part locations. Although using keypoint locations enables a more rigorous evaluation of memory, it also raises a few concerns: does the order of fixations matter? If so, how to order the fixations? How should each each fixation be described? This section explores these questions and proposed a method for investigating different functional aspects of memory in active vision systems.

As the available hardware resources are always finite, every active vision system's memory has to have a limited bandwidth. As a result, during prolonged interactions with a scene the system is bound to reach a point where it has to choose between either preserving the previously memorized information or writing the newly observed information into memory. Determining the correct decision in this scenario requires insight into the relative contribution, or value, of the information stored in memory as compared to the information captured in the new observation. The information that is more valuable to a successful completion of the task at hand should be prioritized.

This tension between preserving memorized information and capturing new information could hypothetically manifest itself in scenarios other than a prolonged interaction with the scene. For example, if all of the scene's valuable information was captured during the first timestep then the system should ideally ignore all information captured during any subsequent observations in order to not corrupt the representation stored in its memory. Conversely, if the system has only been exposed to low value information before its final high value observation then at the final timestep it should overwrite its memory contents with the newly observed information. These scenarios demonstrate the importance of the observations' relative values; they also suggest that controlling the order of the observations can be used to gain additional insight into the functioning of the system's memory.

The first step of the proposed method for evaluating memory by controlling observation value is the implementation of a number of hardcoded attention policies, each one capable of generating a single observation location for every image in the dataset. “Hardcoded“ in this case refers to a policy that is not learnable. In this project this is accomplished using prioritized lists of the anatomical part locations that are provided with the Birds dataset, but those policies can utilize non-learnable algorithms such as corner detection. The locations produced by the different hardcoded attention policies should consist of as few duplicates and have as little overlap as possible, since that can lead to the active vision model re-visiting previously observed areas of the scene instead of sampling new information.

After their implementation, the policies are evaluated by measuring their contributions to the successful completion of the active vision model’s visual task. This can be done by training an *appraisal model* with the observations that the active vision system would be exposed to if it utilized the attention policy being evaluated, i.e. observations processed by the same visual sensor that is employed by the active vision system. The *appraisal model* is a passive vision feedforward model trained to classify images resulting from a given attention policy. Since each policy only produces one fixation per image the appraisal model does not need to utilize active vision and has no need for recurrence, memory or any aggregation strategy at this stage. The appraisal model’s peak classification accuracy on the validation set is utilized as an estimate of the value of the fixation policy that was used to train it. The true value of each hardcoded attention policy cannot be found in this manner as it would require a perfect appraisal model capable of extracting all valuable information out of every observation. Instead of finding the true value, the effective value of each hardcoded attention policy can be estimated if the appraisal model consists of the active vision system’s feature extractor appended with its mechanism for completing the visual task, i.e. a fully connected layer for image classification. The term “effective“ in effective value refers to relativizing the policies’ values to the active vision system’s architectural components, which is advantageous as it prevents the system’s feature extraction component from confounding the results of experiments that focus on memory.

The final step to the proposed method is to test specific functional aspects of the memory mechanism under investigation by selecting the appropriate sequences of hardcoded attention policies. This project identifies three functional aspects: preserving feature representations, forgetting and ignoring. Memory’s capacity to preserve the representations produced by the feature extractor should be tested by seeing if the active vision system’s performance matches that of the appraisal model when using a high value policy.[1] This functional aspect of memory does not require any sequencing of attention policies, but it should be tested in the high performance regime in order to test whether memory acts as a bottleneck to performance even during single time step interactions. Memory’s ability to ignore refers to suppressing

[1]As a reminder, a high value attention policy is one that enabled the appraisal model to obtain a high classification accuracy on the validation set.

a low value input and preserving valuable memorized information; it can be stress-tested by initializing the network with a location chosen by a high value policy and then observing how many low value policies does it take for its performance to degrade below a pre-defined threshold. Forgetting is memory's ability to uptake a valuable observation at the expense of memorized information that is of lesser value, and it can be tested by evaluating the network's performance with a number of low value attention policies followed by a high value policy; a large gap between the resulting performance and the appraised score of the high value policy would indicate the network's lacking ability to forget.

It is easy to assume that only two hardcoded attention policies are required for a basic evaluation of forgetting and ignoring: a low value policy and a high value policy. Although a $[high, low]$ sequence would be adequate for testing ignoring, evaluating forgetting with a $[low, high]$ sequence would be prone to being confounded by the network's ability to ignore, as it may choose to never take up the first observation. This issue can be addressed by replacing the low value policy with one that provides medium value, as that will more likely result in memory taking up some information in the first time step. As a result two hardcoded attention policy sequences are required: $[medium, high]$ for investigating forgetting, and $[high, low]$ for investigating ignoring. As the second policy in the first sequence and the first policy in the second sequence are both $high$, the two sequences can be combined to form a $[medium, high, low]$ sequence, which, if evaluated at every time step, can be used to simultaneously test both the forgetting and ignoring aspects of memory. Although more rigorous stress testing of the different aspects of memory can be achieved with multiple policy sequences, only the $[medium, high, low]$ sequence is used in this project in order to simplify and accelerate experiments.

There exists a way to extend the proposed method and make it more rigorous. It involves using an appraisal model capable of aggregating multiple observations in a single forward pass; several such models are described and used in Section 4.4. The aggregating appraisal model can be used to estimate the mutual information, or mutual value, between the different attention policies by evaluating their combined values.

Suppose that we have two policies $\{p_1, p_2\}$ with their respective individual values[2] being 0.45 and 0.65, and that the aggregating appraisal model estimates their combined value to be 0.72. This would indicate that $0.72 - 0.65 = 0.07$ of $p_1$'s value is unique as it has only contributed that much of a performance improvement over just using $p_2$. It would also indicate that $p_1$ shares $0.45 - 0.07 = 0.38$ of its value with $p_2$. This method can be chained across multiple policies and used to control the redundancy between observations in order to test memory's ability to aggregate new information in different contexts, such as with a nearly saturated memory or with a low value observation. Moreover, uncontrolled

[2]"Value" meaning classification accuracy on the validation set from the appraisal model.

information overlap has the potential to complicate evaluations of active vision memory's capacity to ignore, as discarding low value information is a distinct scenario from discarding redundant information.

The proposed method for Controlling Observation Value, henceforth abbreviated with *COV*, only applies to evaluating functional aspects of memory with respect to the system's visual task. Choosing where to look is likely to have unique requirements that need to be accounted for, which is why this method is not adequate for evaluating the suitability of a memory implementation for the learning of an active vision policy.

## 4.3 Selecting and Appraising Attention Policies

### 4.3.1 Motivation and Objectives

Section 4.2 has described the COV method for investigating memory in general terms, whereas this section applies the first step of the method within the context of the Birds dataset; this first step is the selection and appraisal of a number of hardcoded attention policies on the Birds dataset. Appraised attention policies are instrumental in conducting an interpretable evaluation of memory as they enable an assessment of memory's capacity to preserve useful representations, ignore and forget. The objective of the experiments conducted in this section is to provide the information needed to select a $[medium, high, low]$ hardcoded attention policy sequence that will be used in latter experiments in this chapter.

### 4.3.2 Materials and Methodology

| ID | Anatomical Part Location |
|---|---|
| 1 | Back |
| 2 | Beak |
| 3 | Belly |
| 4 | Breast |
| 5 | Crown |
| 6 | Forehead |
| 7 | Left eye |
| 8 | Left leg |
| 9 | Left wing |
| 10 | Nape |
| 11 | Right eye |
| 12 | Right leg |
| 13 | Right wing |
| 14 | Tail |
| 15 | Throat |

Table 4.1: The anatomical part locations and their respective IDs in the Birds dataset.

All experiments in this section were conducted using Pytorch 1.10.0+cu102 on a PC running Ubuntu 18.04 with a GeForce GTX 1080 Ti and an Intel Core i7-7700. The visual task is bird species classification on the Caltech-UCSD Birds 200-2011 dataset [61] that has been previously described in Section 3.3.2.

The experiments consist of training an appraisal model on individual hardcoded attention policies in order to attribute a value to each one, expressed in terms of peak validation accuracy achieved during training. The appraisal model is a ResNet18 [47] that has been pretrained on the Imagenet dataset [30], however its final fully connected layer has been replaced with a randomly initialized fully connected layer that has 200 output units with the log softmax function applied to them. The inputs to the model are 37x37*px* foveal image patches from the crude retina.

Training was set to run for up to 100 epochs, with early stopping set to end training if validation accuracy did not improve for at least 0.5% over a period of 40 epochs. The batch size was set to 16, the optimizer used was SGD with momentum set to 0.9 and weight decay set to $5.0e^{-3}$. The classification loss is computed using negative log likelihood. The initial learning rate was set to $5.0e^{-3}$ and a scheduler was employed that cut it in half every 10

| **First Location** | **Priorized Lists of Anatomical Part Locations** | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Left eye | 7 | 11 | 6 | 5 | 15 | 10 | 2 | 4 | 3 | 1 | 9 | 13 | 8 | 12 | 14 |
| Left leg | 8 | 12 | 3 | 4 | 15 | 9 | 13 | 1 | 10 | 5 | 6 | 7 | 11 | 14 | 2 |
| Breast | 4 | 3 | 15 | 8 | 12 | 1 | 2 | 6 | 7 | 11 | 5 | 10 | 13 | 9 | 14 |
| Beak | 2 | 15 | 6 | 11 | 7 | 5 | 10 | 4 | 3 | 1 | 8 | 12 | 9 | 13 | 14 |
| Tail | 14 | 9 | 13 | 1 | 10 | 5 | 15 | 3 | 4 | 6 | 2 | 12 | 8 | 11 | 7 |

Table 4.2: The hardcoded attention policies consisting of prioritized lists of anatomical part locations. Each policy is named by its first location, visible in the left-most column.

epochs. The RNG seed for data sampling was set to 303, whereas the seed for the other aspects of training was set to 1, 9 and 9001.

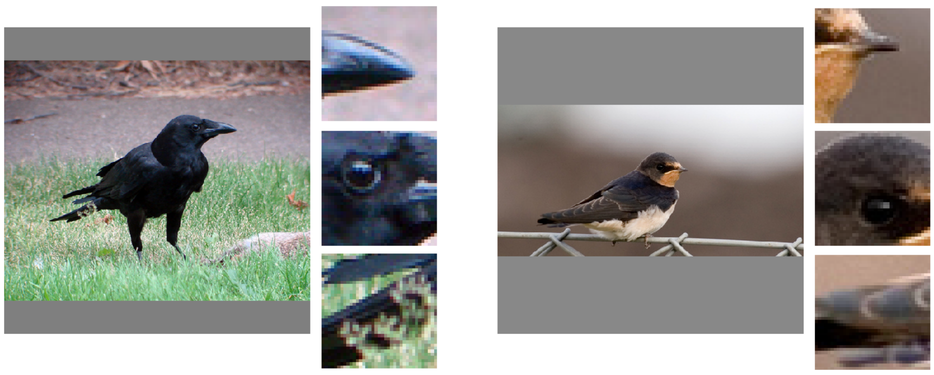


Figure 4.1: Sample input images and their corresponding foveal images produced from the selected $[medium, high, low]$ attention policy sequence: $[Beak, LeftEye, Tail]$.

All images in the Birds dataset are labelled with up to 15 anatomical part locations. The names and the respective IDs of the different anatomical part locations can be found in Table 4.1. The reason some images do not have all 15 locations labelled is due to occlusions: in cases where a body part was occluded the authors of the dataset decided to omit labels instead of approximating the body part's location. This makes individual part locations not suitable as attention policies, as they would not produce any observations for many images. Instead, the 5 hardcoded attention policies shown in Table 4.2 are devised; they consist of prioritized lists of all possible anatomical part locations, making sure that an observation is produced for every image. Each list starts with a different part location, and the locations that follow it are approximately ordered by their anatomical distance to the first one. All part locations have normal noise with $std = 4$ applied to them; the locations are then rounded to the nearest integer with a "round half to even" caveat. Each of the resulting policies is henceforth referred to with the name of its first part location; they can be found in Table 4.2.

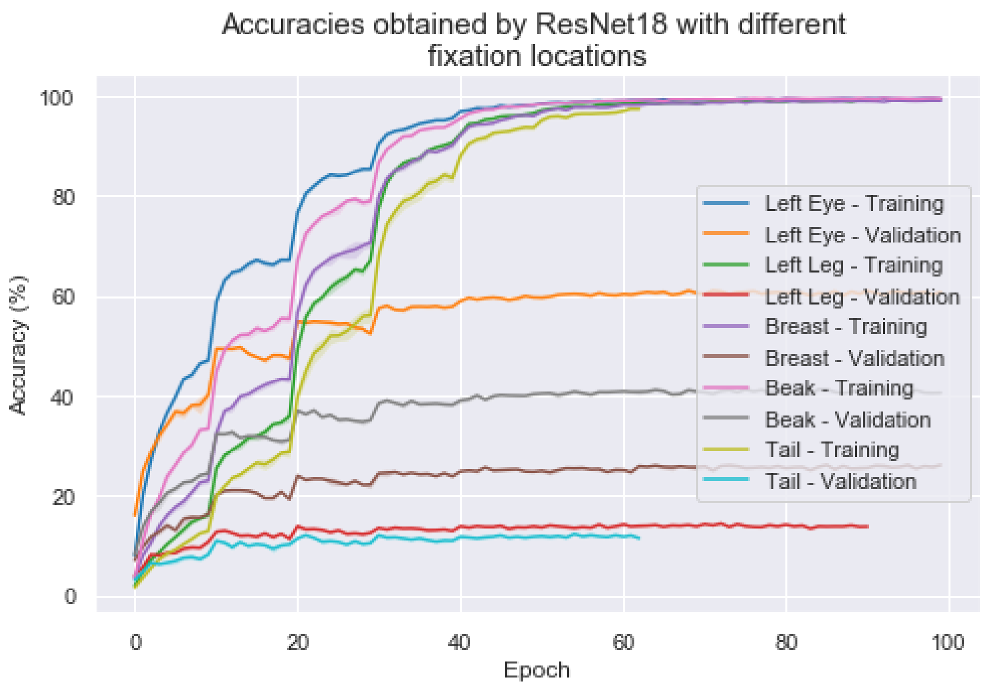


Figure 4.2: Accuracies of the appraisal model when trained with different hardcoded attention policies. Lines represent mean values, shaded regions represent the standard deviation.

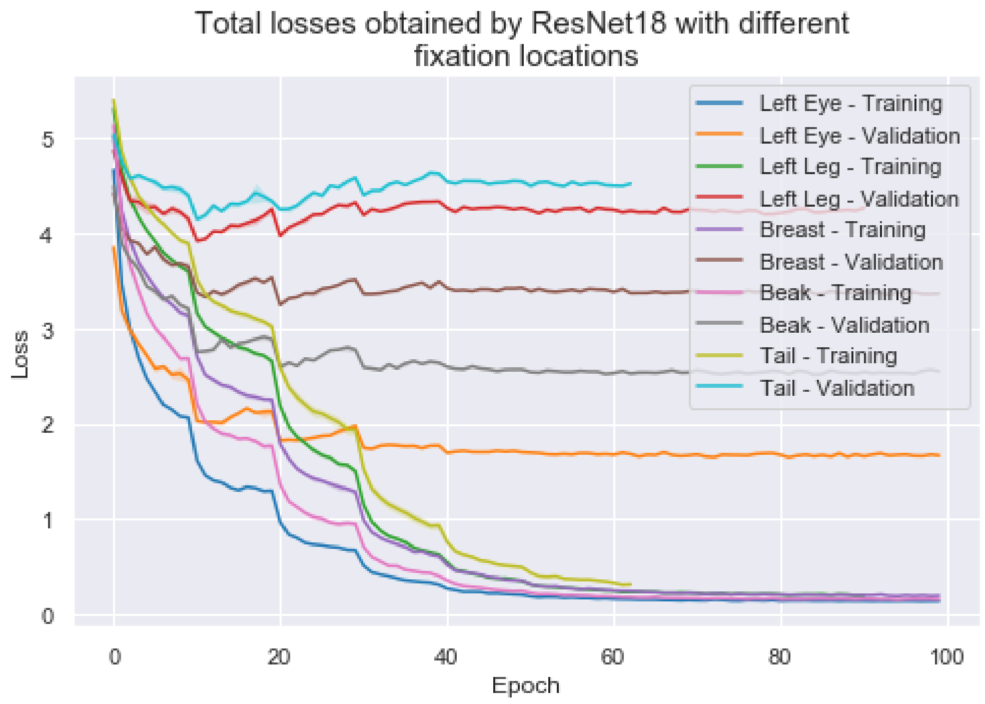


Figure 4.3: Losses of the appraisal model when trained with different hardcoded attention policies. Lines represent mean values, shaded regions represent the standard deviation.

### 4.3.3 Results and Discussion

All attention policies tested have led to the appraisal model approaching $100\%$ accuracy on the training set, as seen in Figure 4.2. The losses showed no anomalies and closely matched the accuracies, as seen in Figure 4.3. The policies that were the closest to the birds' heads, i.e. left eye and beak, have proven to be the most valuable. This was most likely due to the fact that birds' heads contain the most discriminative features. Their tails and legs proved to be the least valuable, which can probably be attributed in part to a lack of discriminative features and in part due to the fact that it is rare for a photograph to be taken from an angle that successfully captures any features present on a bird's legs or tail. Table 4.3 shows the values attributed to each attention policy, which correspond to the appraisal model's classification accuracy on the validation set. The attention policies $[Beak, LeftEye, Tail]$ were selected in order to produce a $[medium, high, low]$ value policy sequence to be used in latter experiments.

| Attention Policy Name | Value (peak validation accuracy) |
|---|---|
| Left eye | 0.61 |
| Left leg | 0.14 |
| Breast | 0.26 |
| Beak | 0.41 |
| Tail | 0.12 |

Table 4.3: The appraised values for different attention policies.

## 4.4 Feedforward Aggregation Methods

### 4.4.1 Motivation and Objectives

The COV memory investigative method proposed in Section 4.2 together with the attention policy values obtained in Section 4.3 can help evaluate memory's ability to preserve feature representations, as well as forget and ignore information that is not needed. One limitation of the proposed method is that it does not measure memory's ability to extract additional features that exist across observations. In other words, it does not measure memory's ability to aggregate multiple observations in a way that effectively combines their values in a cumulative manner instead of merely memorizing valuable information at the expense of less valuable yet still useful information.

In order to address this gap, this section aims to provide points of reference that the different memory implementations will be compared against. The experiments in this section evaluate a number of appraisal models that implement different feedforward aggregation methods. These models serve as a means of appraising the combined value of the selected sequence of attention policies. They are relatively simple solutions to the problem of memory that, admittedly, are not always efficient for various reasons, but are nonetheless valuable in studying active vision memory. The experiments conducted in this section aim to inform future active vision memory development by reviewing different feedforward aggregation methods and assessing their functional characteristics.

### 4.4.2 Materials and Methodology

All experiments in this section were conducted using Pytorch 1.10.0+cu102 on a PC running Ubuntu 18.04 with a GeForce GTX 1080 Ti and an Intel Core i7-7700. The visual task is bird species classification on the Caltech-UCSD Birds 200-2011 dataset [61] that has been previously described in Section 3.3.2.

The experiments in this section compare 5 different feedforward aggregation strategies against each other by measuring their losses and accuracies when trained with the $[Beak, LeftEye, Tail]$ sequence of attention policies. All of the strategies utilize broadly the same architecture that was described in Section 4.3.2: a ResNet18 [47] that has been pretrained on the Imagenet dataset [30] and that has a randomly initialized fully connected layer with 200 log softmax output units.

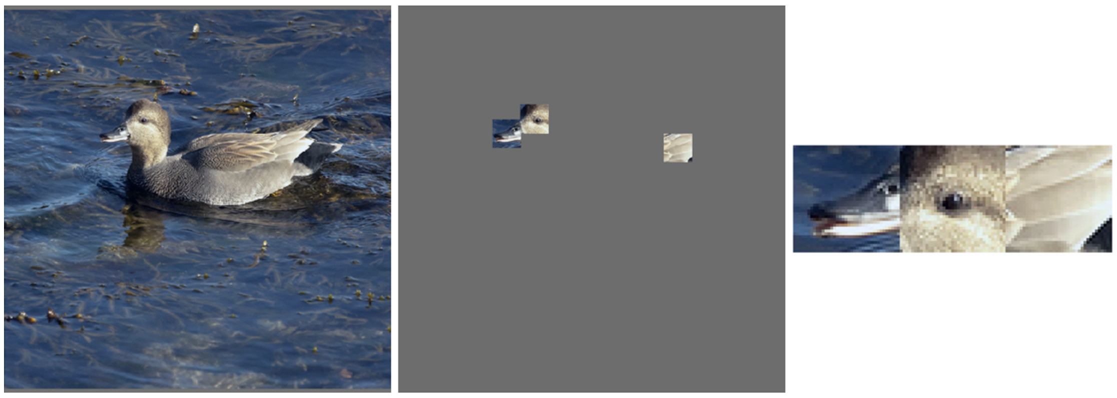


Figure 4.4: A visualisation of two aggregation strategies using the $[Beak, LeftEye, Tail]$ sequence of attention policies. **Left:** original input image. **Center:** the input image produced by the unmasking strategy. **Right:** the input image produced by the imagespace concatenation strategy, scaled up for visibility.

The 5 aggregation strategies proposed in this section are *unmasking*, *spatial concatenation*, *feature averaging*, *output (softmax) averaging* and *output (pre-softmax) averaging*. Under *unmasking* the model receives as input the entire input image with all of the unobserved areas masked out with 0s and with only three 37x37*px* areas shown unmasked in full resolution, corresponding to the observations provided by the attention policies. This large input size does not lead to an explosion in the number of network parameters as an adaptive average pooling layer is applied to the latent space prior to the fully connected layer. Prior to conducting the experiments the unmasking strategy was hypothesized to be superior as it expresses the relative locations of the different observations made by each attention policy. The *spatial concatenation* strategy refers to feeding the appraisal model with an input space concatenation of the 37x37$px$ image patches, resulting in a 37x111$px$ input size. A visualisation of the unmasking and imagespace concatenation strategies can be seen in Figure 4.4. *Feature averaging* refers to combining multiple observations by averaging the latent space tensors produced by the ResNet18 network before applying adaptive average pooling. The intuition behind it is that the latent space tensors code for the presence of class-specific features in the image, and averaging them will result in averaging the amount of evidence present for each class. Both cases of *output averaging* refer to averaging the classification output of the network, in one case prior to applying the log softmax activation function and in the second case after its application.

The training parameters used in this section match those described in Section 4.3.2. The maximum number of epochs was set to a 100, with early stopping set to trigger if validation accuracy did not improve by 0.5$\%$ or more over 40 epochs. The batch size was set to 16, the optimizer used was SGD with momentum set to 0.9 and weight decay set to $5.0e^{-3}$. The classification loss used is negative log likelihood. The initial learning rate was set to $5.0e^{-3}$ and a scheduler reduced it by half every 10 epochs. The RNG seed for data sampling was set to 303, and the seed for the other aspects of training was set to 1, 9 and 9001. All part locations have normal noise with $std = 4$ applied to them; the locations are then rounded to the nearest integer with a “round half to even“ caveat.

### 4.4.3 Results and Discussion

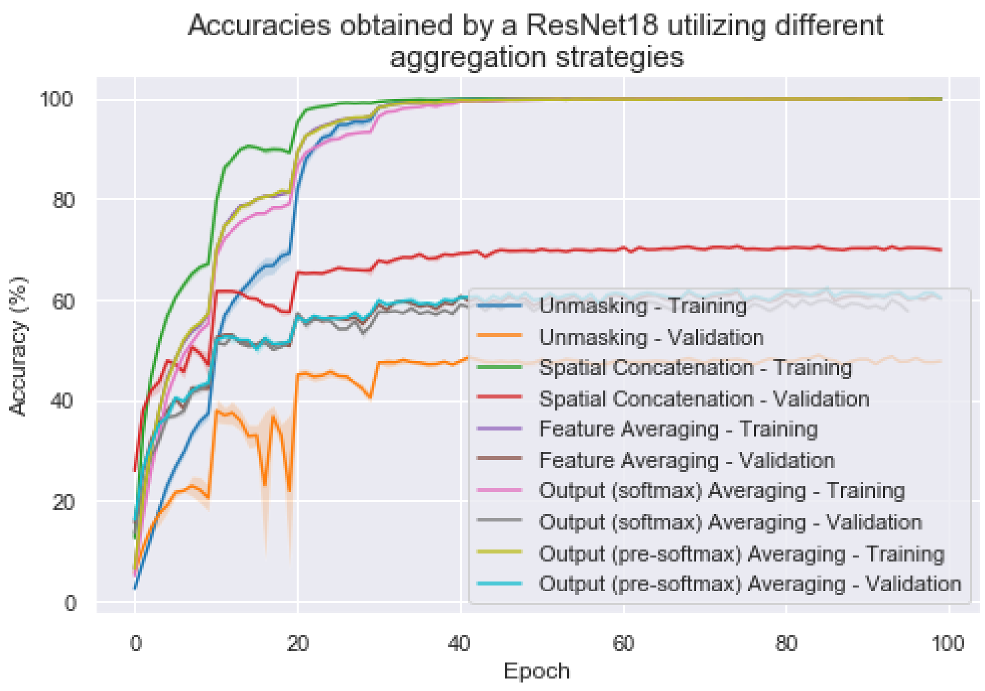


Figure 4.5: Accuracies of the appraisal model when trained with different aggregation strategies. Lines represent mean values, shaded regions represent the standard deviation.

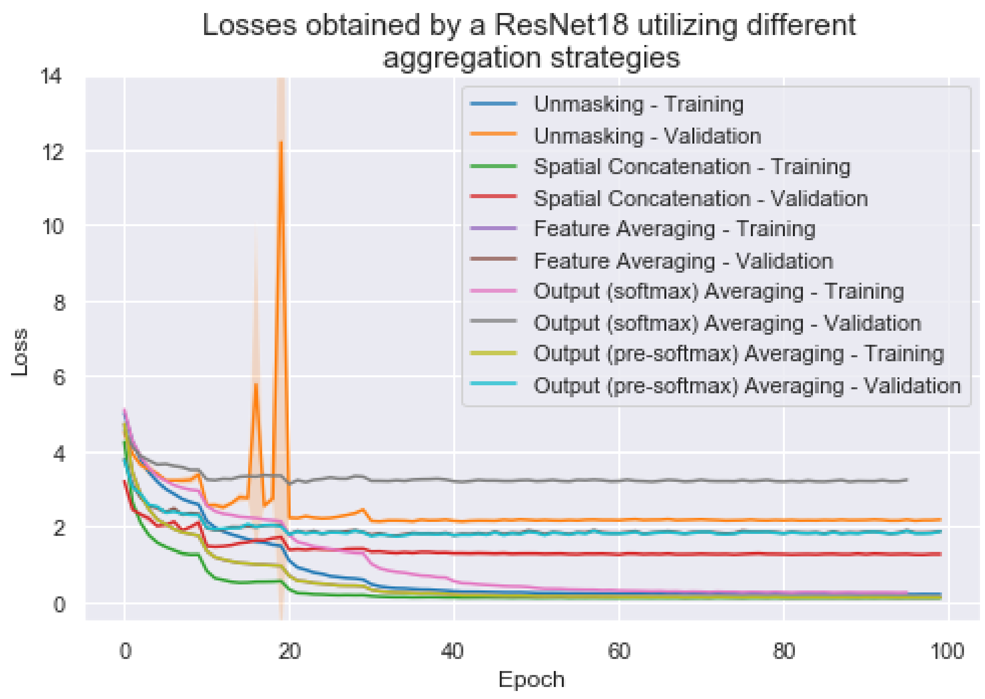


Figure 4.6: Losses of the appraisal model when trained with different aggregation strategies. Lines represent mean values, shaded regions represent the standard deviation.

The spatial concatenation strategy has shown superior performance in terms of both its validation accuracy and validation loss. This is likely due to the fact that with this strategy the network utilized the large receptive field size of the deeper convolutional layers of ResNet18 to extract features from across multiple observations. The experiments conducted in this section can be interpreted as testing the merging of multiple observations at different locations in a feedforward classification model, and this result suggests that merging them in the image space is preferable to doing so deeper in the model. Table 4.4 contains the peak validation accuracies achieved by the different strategies; it shows that this spatial concatenation outperformed the averaging strategies by 9-11%. One reason for its better performance could be due to the fact that averaging multiple observations can be destructive to the information that they contain individually.

Judging by validation accuracy alone, the unmasking strategy has the worst performance out of all the strategies tested, however the loss plots show that output averaging after the softmax activation function has resulted in a significantly worse validation loss. The poor performance of unmasking is likely due to the fact that the input image is quite sparse, which in turn leads to a sparse output of the convolutional feature extraction of ResNet18; this sparsity can then cause the adaptive average pooling layer to output zeros or values with a really low magnitude, causing numerical issues. The noisy outliers visible in epochs 15 to 20 in the loss and accuracy plots support the idea of numerical issues by suggesting instabilities during training. Several possible ways of addressing this issue are regularizing or normalizing the feature maps before applying adaptive average pooling, only applying average pooling to non-zero values, or resorting to maxpooling, although maxpooling would result in sparser gradient updates. The reason for the softmax output averaging strategy's high validation loss in spite of it having better validation accuracy is likely the poor calibration in the network's outputs, which can mean either over-confidence during incorrect predictions or under-confidence during correct predictions. Pre-softmax and feature averaging strategies have turned out to be roughly equivalent in their performance.

| **Aggregation Strategy** | **Peak Validation Accuracy** |
|---|---|
| Unmasking | 0.49 |
| Spatial Concatenation | 0.71 |
| Feature Averaging | 0.62 |
| Output (softmax) Averaging | 0.60 |
| Output (pre-softmax) Averaging | 0.62 |

Table 4.4: Top validation accuracies for different aggregation strategies.

These results provide a valuable range of values that can be used as points of reference for evaluating different recurrent memory implementations. They also suggest that the convolution operator can be leveraged for extracting features across multiple observations if these are represented appropriately, and that care has to be taken to avoid causing numerical issues within the feature space.

## 4.5 Temporal Robustness

### 4.5.1 Motivation and Objectives

The COV method for investigating memory described in Section 4.2 involves evaluating the active vision system's performance after being exposed to a varying number of observations. This could be achieved with a single instance of the model if it was temporally robust. In this thesis a model is considered temporally robust when its performance is not negatively affected by changing the time step at which it produces a classification output. The literature has not investigated any training methods that might be capable of guaranteeing such a functional characteristic in sufficient depth. Curriculum learning and intermediate supervision are two memory-specific training methods that hypothetically could help and that have been touched upon in the investigation by Li et. al. [12]; however, as previously mentioned in Section 3.2.2, their experimental setting trivialized active vision in a way that casts doubt as to whether their results generalize to active vision problems. Moreover, an examination of the GitHub repository linked in the abstract of the paper reveals that the authors have instantiated a separate fully connected classification layer for each time step that their model interacts with the scene. In spite of this being a significant aspect of their architecture that impacts the transferability and implications of their approach, the authors did not describe it in their publication.

Even if a different set of model weights has to be loaded at each timestep, the network's memory can still be evaluated in the same way with the assumption that the results produced by the multiple instances approximate those that would be produced by a single instance trained with the appropriate method. Searching for such a training method falls within the scope of this project and is the focus of this section. An active vision model that is invariant to the time step at which it produces a classification output is more temporally robust as it can handle a wider variety of interaction scenarios. This section investigates a number of different methods focused on memory function and the temporal aspect of active vision. The objectives are to find out which of the tested methods is best for ensuring temporal robustness and which one is suitable for investigating memory alongside the COV method.

### 4.5.2 Materials and Methodology

All experiments in this section were conducted using Pytorch 1.10.0+cu102 on a PC running Ubuntu 18.04 with a GeForce GTX 1080 Ti and an Intel Core i7-7700. The visual task is bird species classification on the Caltech-UCSD Birds 200-2011 dataset [61] that has been previously described in Section 3.3.2.

The experiments in this section compare 5 different approaches to ensuring temporal robustness. One of the objectives of this section is to find a method for ensuring temporal robustness that is most suitable for use alongside the COV method, therefore some of the approaches tested suffer from technical limitations that make them unsuitable for use outside of research. The different approaches are all applied to a modified recurrent attention model that is a simplified variant of the model used in Section 3.3.2. The model utilizes a single foveal $37x37px$ image patch as its sensor and a ResNet18 [47] pre-loaded with ImageNet weights as its feature extractor, with its final fully connected layer removed. The 512 channel output of ResNet18's adaptive average pooling layer feeds an LSTM network that maintains hidden state and memory tensors, both of which are 512 units large and both of which are initialized to zeros at the beginning of each interaction sequence. The hidden state tensor is used as input to a fully connected classifier layer that has 200 log softmax output units. Unlike the model in Section 3.3.2, this model does not include a proprioception stream, an attention module or a baseline module.

The five approaches tested are *single instance*, *curriculum learning*, *intermediate supervision*, *BN-LSTM*, and *multiple instances*. *Single instance* corresponds to regular neural network training, *curriculum learning* and *intermediate supervision* have been previously used in a different setting by [12], BN-LSTM was described in [96] and [97], and *multiple instances* has not been explored in prior literature. None of these methods have been previously evaluated with regards to their impact on temporal robustness, as this concept is only being introduced in this chapter.

Under *single instance* the modified recurrent attention model is trained once on the full sequence of hardcoded attention policies before being evaluated three times, once at every time step of the policy sequence, meaning after having observed $[medium]$, $[medium, high]$, and $[medium, high, low]$. This approach is the simplest and has the advantage of not requiring a prolonged training process or multiple model instances, but it has no explicit mechanism for ensuring temporal robustness. **The author's hypothesis is that the single instance approach will fail to achieve temporal robustness because the LSTM's hidden state and memory tensors are produced by a network with a different effective depth at every time step.** This can cause these tensors to have different distributions with different properties that the fully connected classifier network can not handle due to only having been trained on memory tensors produced after three time steps.

*Curriculum learning* aims to alleviate this problem by gradually increasing the number of time steps during training; this is achieved by training a randomly initialized model with one time step, using the resulting model weights to initialize training with two time steps, and then initializing the three time step training run with the final weights of the previous two time step run. It can be understood as training a deeper network on top of the weights of a shallower one. **The hypothesis motivating the use of the curriculum learning method is that the lower time step functionality might persist through further training owing to the curriculum initialization scheme, improving temporal robustness.** The method could be further fine-tuned by only selectively preserving different parts of the model across time steps, but this approach is left for future work.

The *intermediate supervision* approach refers to training a single instance of the recurrent attention model on the full policy sequence, requiring it to output a classification at every time step and including all of the resulting classification outputs in the loss function. This is also known as the greedy loss approach. **The hypothesis motivating intermediate supervision is that the multiple sources of classification gradients will result in the model hosting multiple subnetworks, each one specialized in classifying at a different time step.**

*BN-LSTM* refers to applying the single instance approach to a batch normalized LSTM [97] [96], which is an LSTM variant with a separate batchnorm layer applied to its memory tensors at every time step. **The motivating hypothesis behind using the BN-LSTM is that the batchnorm layers might help regularize the LSTM's memory tensors across time, neutralising the negative effect of variable effective depth and making the intermediate time step tensor more suitable for the fully connected classifier layer.**

*Multiple instances* refers to the previously mentioned idea of training a separate model instance for each time step in the policy sequence. **The hypothesis motivating the use of this approach for research is that the results will approximate those that would be produced by a single instance trained with the appropriate method, if such a method exists.** This method is not suitable for ensuring temporal robustness in non-research settings due to the need to load different model weights. The five approaches listed were not evaluated in tandem with each other.

In these experiments, the available observations were shuffled during training so as to not bias the network to associate inputs at specific values with specific time steps, i.e. to prevent the model from always expecting a high value observation at $t = 2$. At validation time the observations were kept in the $[medium, high, low]$ order, although the full sequence was only available during the third run. Besides that, the training parameters used match those used during previous experiments in Section 4.3.2 and Section 4.4.2. The maximum number of epochs was set to 100, with early stopping set to trigger if validation accuracy did not improve by 0.5% or more over 40 epochs. The batch size was set to 16, the optimizer used was SGD with momentum set to 0.9 and weight decay set to $5.0e^{-3}$. The classification loss used is negative log likelihood, with the *intermediate supervision* variant computing an average of the loss across time steps. For this reason the loss produced by this variant is significantly different from that of the other variants. The initial learning rate was set to $5.0e^{-3}$ and a scheduler reduced it by half every 10 epochs. The RNG seed for data sampling was set to 303, and the seed for the other aspects of training was set to 9001. All part locations have normal noise with $std = 4$ applied to them; the locations are then rounded to the nearest integer with a “round half to even“ caveat.

### 4.5.3 Results and Discussion

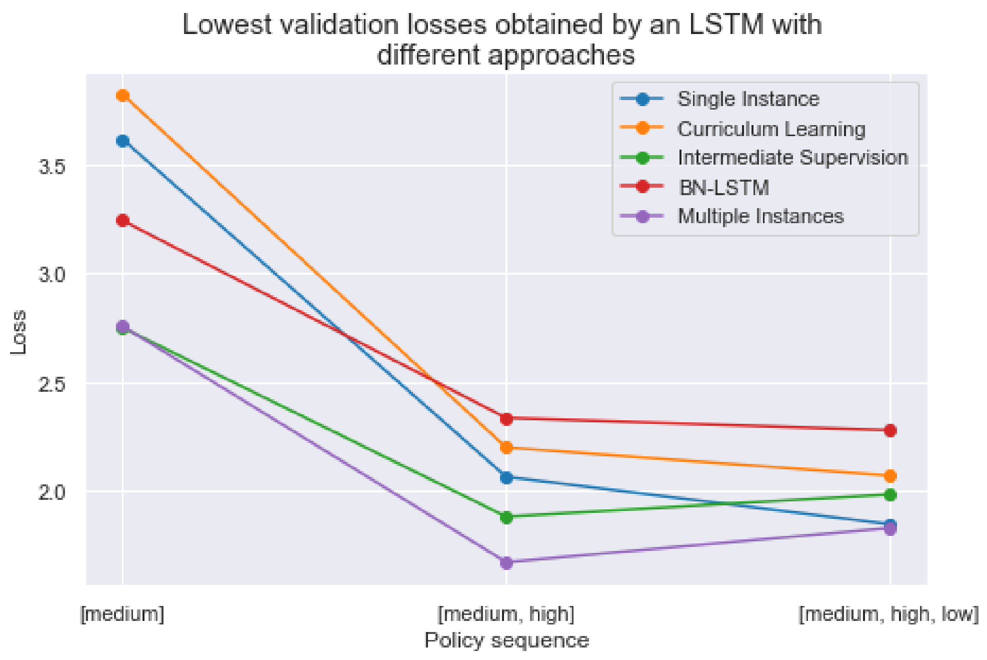


Figure 4.7: Best validation losses of models using different approaches for promoting temporal robustness. Each data point represents the best result achieved over all training epochs with the model trained on a different policy sequence. A latter review of this work revealed that a superior approach in this figure would involve taking the mean of N best performing epochs. Such an analysis would prove more robust to noisy classification losses.

Figures 4.7 and 4.8 show the policy sequence plots that visualize the results of evaluating the different approaches with the selected attention policy sequence. It is worth noting that the loss of the *intermediate supervision* approach is computed slightly differently than the other losses on the plot, as it is the average of the losses resulting from the model’s classifications at every time step. Both of the plots also only include the validation values and omit the training values as the training values would not be informative due to the best values being selected for each of the three data points. The accuracy plot also includes the individual policy values and the highest appraised combined value for reference.

Although an attempt could be made at interpreting these policy sequence plots to gain insights into memory's functional aspects other than temporal robustness, such as ignoring and forgetting, this analysis is instead done in Section 4.6.3. The reason for this is that only one RNG seed was used in this section, whereas Section 4.6.3 utilizes multiple RNG seeds and compares several memory variants against each other, which lends itself to a better contextualized analysis.

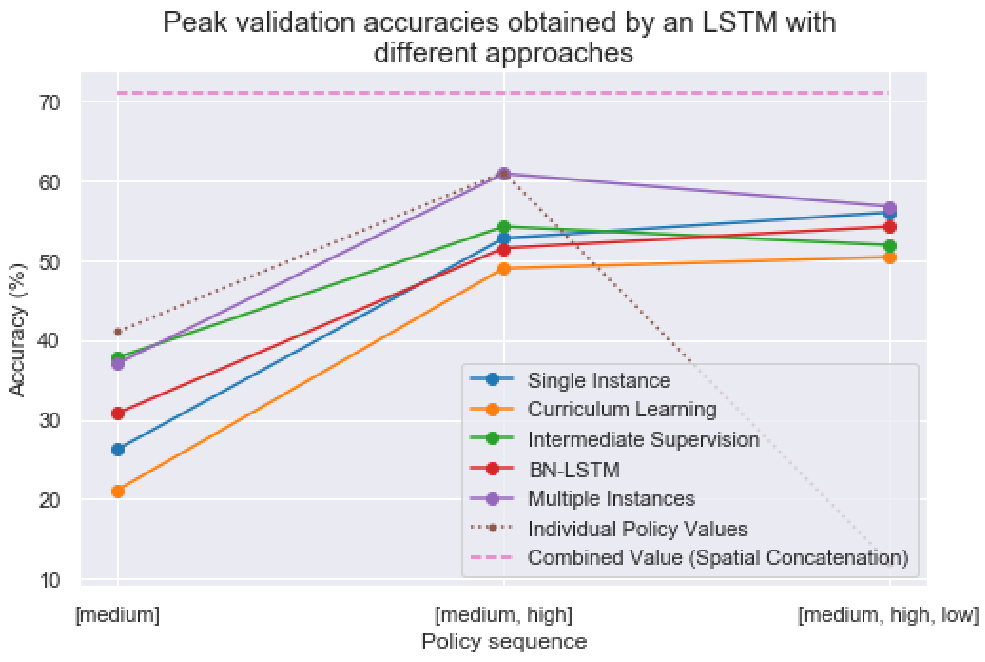


Figure 4.8: Best validation losses of models using different approaches for promoting temporal robustness. Each data point represents the best result achieved during a training run with the model trained on a different policy sequence. The plot also includes the individual policy values obtained in Section 4.3.3 and their peak combined value obtained during experiments in Section 4.4.3

The *multiple instances* approach proved to be the best in terms of both its validation accuracy and loss, and is thus the recommended approach for future investigations. It is not a desirable approach for non-research uses, so it should only be used as a point of reference during further development of active vision memory. By far the worst approach was *curriculum learning*, suggesting that random network weight initialisation is preferable as there are likely little to no features learned during prior training runs that should be preserved for future runs.

The *single instance* approach is the simplest approach in this experiment, and besides multiple instances the only approach that manages to partially improve on it is *intermediate supervision*. With only the first policy the *intermediate supervision* approach was equal in both its loss and accuracy to the *multiple instances* approach; with two policies it improved on the *single instance* approach slightly, but with all of the policies its performance fell significantly. The reason behind this trade-off between the network's performance at the first and the final time steps is likely due to two factors, the first one being that the gradients become weaker as the model's effective depth increases at higher time steps, allowing the gradients from the first time step to dominate. The second reason could be that during the first time step LSTM's memory and hidden state tensors are both zero, and the gradients coming from classification produced at this timestep bias memory towards taking up the observed information at the expense of considering memory contents, as these are blank and the corresponding LSTM gates receive no gradient update. The *intermediate supervision* approach looks the most promising, and a few hypothetical ways of improving its performance could be to initialize memory values randomly at the beginning of each interaction sequence, boost the gradients coming from latter time steps, and exclude the classification made during the first time step from supervision. The *BN-LSTM* approach performed comparably well to the *single instance* approach, showing significant gains with only the first policy and small losses at latter time steps.

These results are yet another example of how the results described in the literature, in this case Li et. al. [12], do not generalize to more strict active vision settings. However, one potential future investigation is to borrow the aspect of their approach that they have not described in their publication and see whether using a separate fully connected classifier layer for every time step of the network can improve temporal robustness. Combinations of these different approaches should also be tested as they may exhibit synergistic effects.

## 4.6 Recurrent Memory Variants

### 4.6.1 Motivation and Objectives

Previous sections have completed all of the preparations needed for comparing different memory variants against each other using the COV method. Section 4.3 has appraised several hardcoded attention policies and combined them into a sequence that acts as a controlled substitute for the attention module; Section 4.4 has evaluated different feedforward aggregation strategies to estimate the combined value of the chosen attention policy sequence; and Section 4.5 has addressed the question of temporal robustness and selected an approach for training the model alongside the COV method. The objective of this section is to make use of the materials prepared in the aforementioned sections and to employ the COV method to evaluate several recurrent memory variants.

Although this has not been discussed previously, the results in Sections 4.4.3 and 4.5.3 reveal that the spatial concatenation strategy for combining multiple observations is superior to the LSTM recurrent memory variant. This casts doubt on a fundamental assumption that structured much of prior active vision research in deep learning, as it questions whether using recurrent memory to aggregate the output of a feature extractor is the right approach. The experiments in this section are aim to find out whether a recurrent memory variant can achieve performance comparable to that of the spatial concatenation aggregation strategy, so as to either motivate or demotivate future research into alternatives to employing recurrent memory in the way the recurrent attention models do.

A secondary objective of this section is to investigate a novel data structure called the WhereWhat matrix, abbreviated with WW. The motivation behind it was to create and investigate a data structure that supports the separation between the content of the image and the location of said content in the image, analogously to the division of the human visual system into the dorsal “where‘ and ventral “what“ streams [98]. This data structure was designed to support future attempts at implementing a dorsal stream for investigating active vision attention; however, these investigations were not conducted in this project and were instead left for future work.

### 4.6.2 Materials and Methodology

All experiments in this section were conducted using Pytorch 1.10.0+cu102 on a PC running Ubuntu 18.04 with a GeForce GTX 1080 Ti and an Intel Core i7-7700. The visual task is bird species classification on the Caltech-UCSD Birds 200-2011 dataset [61] that has been previously described in Section 3.3.2.

The experiments in this section compare four variants of recurrent memory, with two variants operating on input that is of a novel data structure. All aspects of the architecture are kept constant between the different variants tested, aside from the memory network, the operation processing feature maps before passing them to the memory network, and the input size of the fully connected classifier network. As during previous experiments, all models utilize a single 37x37*px* image patch as the sensor and a ResNet18 [47] pre-loaded with ImageNet weights [30] as the feature extractor. All of the variants remove ResNet18’s fully connected layer and the two variants testing the novel data structure replace the preceding adaptive average pooling operation with a novel operation that is described below. The output of the feature extractor feeds the recurrent memory variant, the hidden state of which is used as input to a fully connected classifier layer that has 200 log softmax output units.

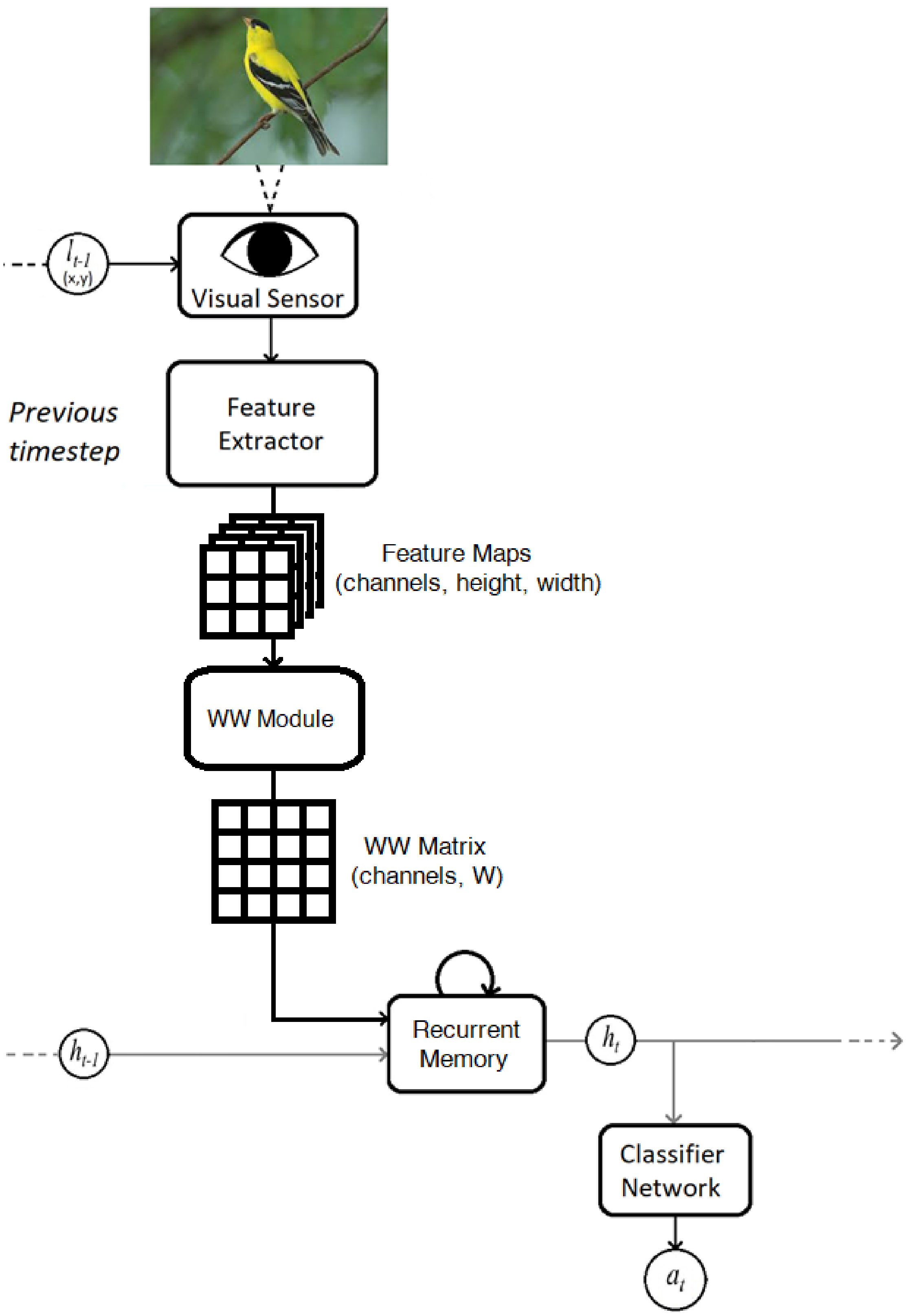


Figure 4.9: A diagram representing the modified recurrent attention model when using the WW memory variants. The memory hidden state tensor $h$ is also a WW matrix. The variable $a$ is the output classification scores and variable $l$ is the image location that the network is fixating upon.

The four recurrent memory variants tested are the *RNN*, the *LSTM*, the *WW-LSTM* and the *WW-RNN*. The gated recurrent unit [62] has not been tested; it can be understood as a compromise between the simple *Elman RNN* and the more complex *LSTM* and thus there is no reason to believe it should behave significantly different from these two memory variants. The *LSTM* network [57] matches the architecture used in Section 4.5, whereas the *RNN* variant implements a simple Elman recurrent neural network [99] with the tanh activation function applied to both the hidden state and the output of the network; both of these variants have their hidden and memory tensor sizes set to 512, which also equals their input size.

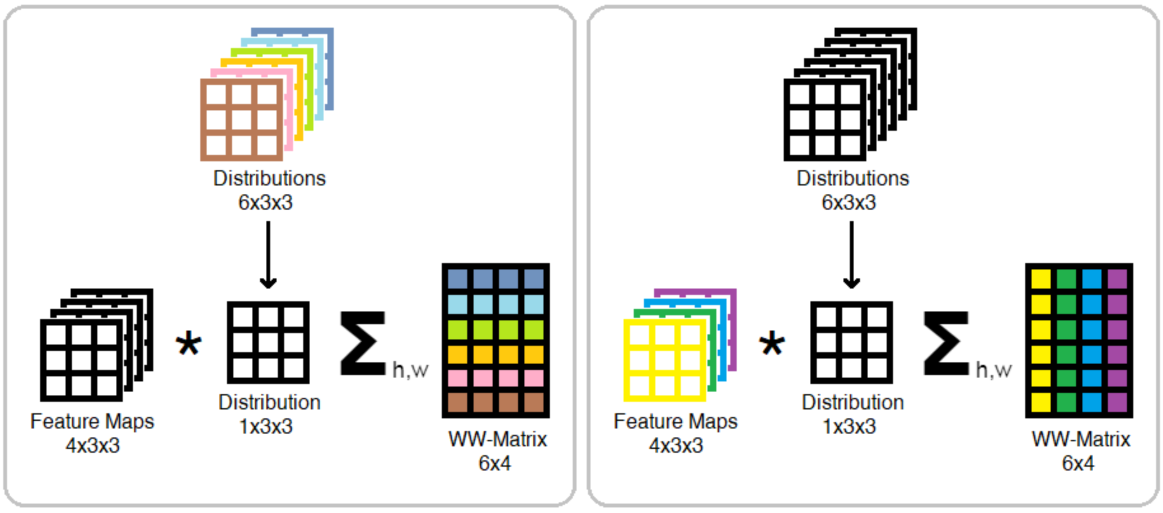


Figure 4.10: Colour-coded diagrams depicting the WW operation applied to a 4x3x3 feature map with $W = 6$ distributions. The distributions scan the feature maps across the channel dimension and their products with the feature maps are spatially summed to produce the WW matrix. **Left:** a unique colour has been applied to each distribution in the WW operation's weight tensor; the corresponding rows in the output WW matrix have been colour coded to match. **Right:** a unique colour has been applied to each channel in the input feature map; the corresponding columns in the output WW matrix have been colour coded to match.

The WW prefix in the *WW-LSTM* and *WW-RNN* memory variants refers to a novel data structure called the WhereWhat matrix. The matrix is produced out of unpooled feature maps, which in this case is a ResNet18. In a manner similar to how the convolution operator slides a kernel with a restricted receptive field across the spatial dimensions of an image input and sums the product of its coefficients and the input image, the WW matrix is produced by sliding a weight tensor across the channel dimension of a feature map input and by summing the product of the weight tensor's coefficients and the input features. The weight tensor of the WW operation has a user-defined number of channels, denoted with $W$, and a set of spatial dimensions equal to the spatial dimensions of the input feature maps; its shape is $(width, height, W)$ whereas the shape of the input is $(channels, height, width)$. The resulting WW matrix has a shape of $(channels, W)$, which is where the “WhereWhat“ name comes from; the “what“ refers to the channels dimension that corresponds to the convolutional feature channels of the input, describing the content of the image. The “where“ refers to the $W$ dimension that corresponds to global spatial distributions of the input, describing where in the field of view are the features located. Figure 4.10 depicts a diagram that uses colour-coding to provide an intuitive understanding of the WW matrix and the operation used to produce it. In this section the $W$ parameter is set to 10.

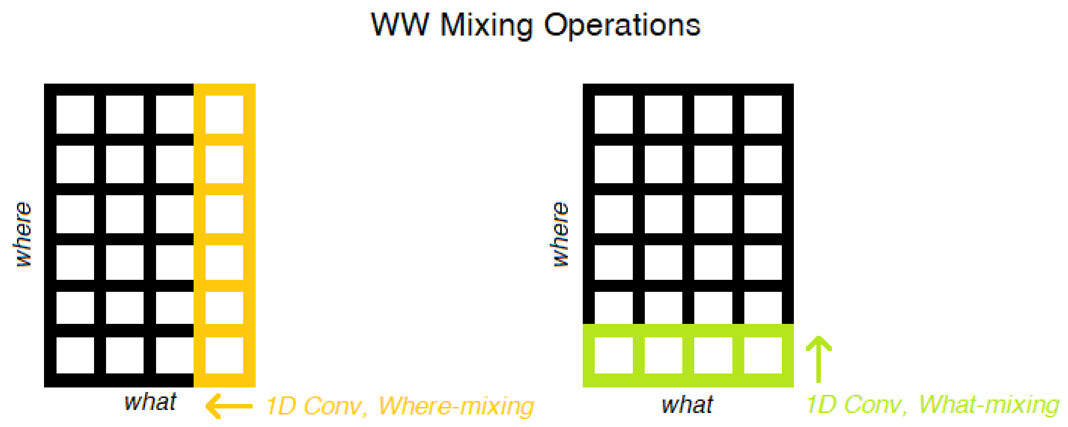


Figure 4.11: A diagram depicting the similarity between WhereMix and WhatMix operations and the 1x1 convolution operation when applied to a 6x4 WW matrix.

One way to understand the WW operation is to view it as a series of $W$ learnable pooling kernels that reduce each feature map of a pre-defined size down to a single value; with $W = 1$ its output size would be equal to that of an adaptive average pooling layer. Alternatively, the WW operation can be understood as a mean of implementing an inductive bias of channel-invariant global feature extraction, just like the convolution operator implements an inductive bias of spatially invariant local feature extraction. The resulting WW matrix can have one of three operations performed on it: it can be flattened and passed to a fully connected layer, or it can be “mixed“ along one of its two dimensions with either a WhereMix or a WhatMix. A WhereMix refers to scanning the WW matrix with a $(1, W_{in}, W_{out})$ kernel along its first $channels$ dimension, summing the product of its coefficients and the matrix elements. A WhatMix refers to scanning the WW matrix with a $(channels, 1, channels_{out})$ kernel along its second $W$ dimension. Each of the mixing operations can be implemented using a 1x1 convolution applied to either canonically oriented or a transposed view of the WW matrix. Figure 4.11 visualizes how the two mixing operations can be implemented with 1x1 convolutions, whereas the code shown in Listing 4.1 implements the WW operation and the two mixing operations.

Listing 4.1: A code excerpt showing an implementation of the WhatMix, WhereMix and the WW operations.

```
import torch
import torch.nn as nn

class WW_module(nn.Module):
    """
    Operation for producing a WW matrix. Expected input shape: (B,
        channels, height, width)

    Args
    - in_shape: input feature maps dimensions, excl. batch dim
    - out_channels: W dim (num of distributions)"""

    def __init__(self, in_shape, out_channels):
        super(WW_module, self).__init__()
        self.W = nn.Parameter(torch.FloatTensor(torch.Size(
                        (1,1,) + in_shape[1:] + (out_channels,))))
        torch.nn.init.xavier_uniform_(self.W)

    def forward(self, x):
        x = x[..., None] * self.W
        return x.sum(dim=(2,3))

class WhatMix(nn.Module):
    def __init__(self, in_shape, out_shape, bias=True):
      super(WhatMix, self).__init__()
      self.conv = nn.Conv1d(in_shape[0], out_shape[0], 1, bias=bias)

   def forward(self, x):
      return self.conv(x)

class WhereMix(nn.Module):
   def __init__(self, in_shape, out_shape, bias=True):
      super(WhereMix, self).__init__()
      self.conv = nn.Conv1d(in_shape[1], out_shape[1], 1, bias=bias)

   def forward(self, x):
      x = self.conv(x.transpose(1,2))
      return x.transpose(1,2) #revert to original orientation
```

The hypothesis motivating the use of the WW data structure is that it will introduce a useful inductive bias into the data processing stream by encoding information about global spatial distributions in the feature maps. In this section only the WhereMix operation has been used as it introduces a much lower parameter count. The fully connected layers in the RNN and in the LSTM's gates have been replaced with the WhereMix operation for these networks' WW variants. The hidden and memory tensors of the *WW-LSTM* and the *WW-RNN* networks are all WW matrices that are flattened before being passed to the fully connected classifier network. This results in a much larger parameter count, but it is significantly off-set by the use of the parameter-efficient WhereMix inside the network. The total model parameter counts in this section are 12,201,592 for the *WW-LSTM* variant, 12,200,972 for the *WW-RNN* variant, 13,378,312 for the *LSTM* variant and 11,804,424 for the *RNN* variant. In the WW variants the majority of parameters are held by the classifier network that has to process a very large hidden state tensor with a fully connected network, whereas the majority of parameters in the non-WW variants are held by the memory networks. No other changes are made to the WW variants of the LSTM and the Elman RNN networks besides replacing the fully connected layers with WhereMixes and changing the memory and hidden tensors' data structure.

The 4 model variants were each trained on the $[medium]$, $[medium, high]$, and *[medium, high, low]* policy sequences, with each training run being repeated over 3 different RNG seeds: 1, 9 and 9001 resulting in a total of 36 training runs. Unlike in previous experiments, this time the RNG seed used in the data sampler was also varied so as to increase the variability between runs. The maximum number of training epochs was set to a 100, with early stopping triggering if validation accuracy did not improve after 40 epochs. The batch size was set to 16, the optimizer used was SGD with momentum set to 0.9 and weight decay set to $5.0e^{-3}$. The initial learning rate was set to $5.0e^{-3}$ and a scheduler cut it in half every 10 epochs. All part locations had normal noise with $std = 4$ applied to them using a separate RNG seed.

### 4.6.3 Results and Discussion

Figures 4.12 and 4.13 show that the different memory variants have performances that are clustered together based on their underlying data structures, with the WW variants peforming similarly to each other and better than the non-WW memory variants. The WW variants exhibit significantly lower training losses, suggesting that these models enjoy more expressive power and a greater ability to fit the data. This could be either due to the WW matrix and its operations contributing useful inductive biases, or it could be due to the increased size of the hidden state tensor and the fully connected classifier layer. In terms of validation accuracy all variants perform similarly well with the $[medium]$ and $[medium, high]$ sequences, with the WW-LSTM edges out the competition.

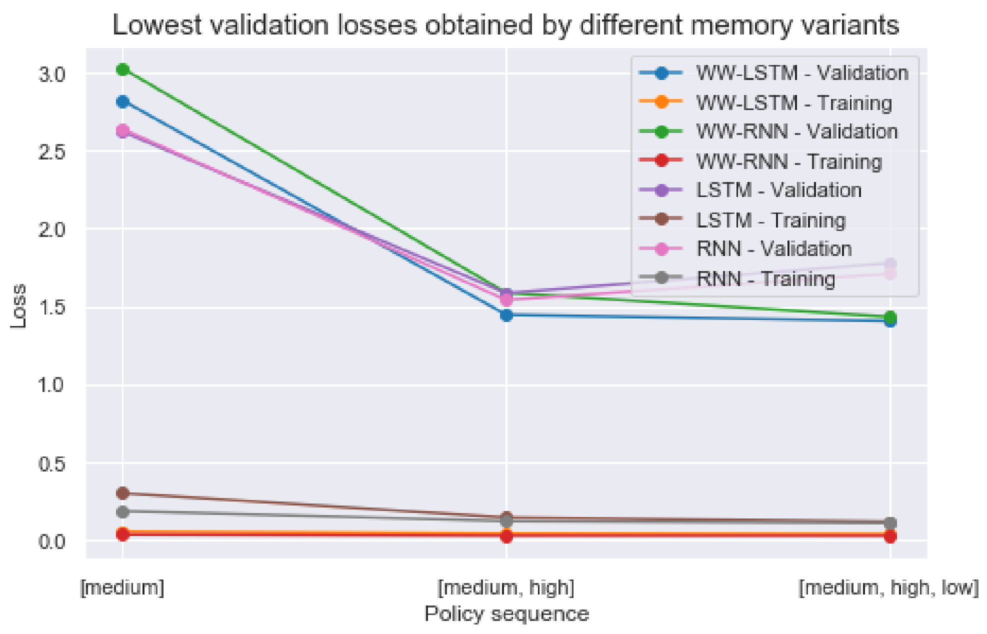


Figure 4.12: Best losses of modified recurrent attention models using different memory variants. Each data point represents the best result achieved during a training run with the model trained on a different policy sequence. Lines represent mean values, shaded regions represent the standard deviation.

With the full policy sequence the WW variants do noticeably better than the vanilla RNN and LSTM. The non-WW memory variants see a performance decrease between the two time step and the three time step sequences, whereas the WW variants see gains instead. This could be due to the WW operations improving memory's ability to ignore the low value observation; however, the mechanism by which this could be happening is not clear and clarifying it is left for future work. Out of all the variants tested the LSTM appears the worst, as it has a low performance and the highest parameter count. The WW variants have comparable parameter counts to the RNN and perform significantly better on the full sequence.

All of the variants tested appear capable of forgetting lower value information in the face of a high value observation and of combining information across observations, as their performance at $[medium, high]$ is consistently higher than that of the individual policy's value. However, there is still a considerable gap in performance between the spatial concatenation strategy and the best performing variant. These results show that although the novel suggested data format offers a significant improvement to recurrent active vision memory, it is not enough to overcome a much simpler aggregation strategy. It is likely that the memory data format has to preserve spatial information of each observation in order to beat this benchmark.

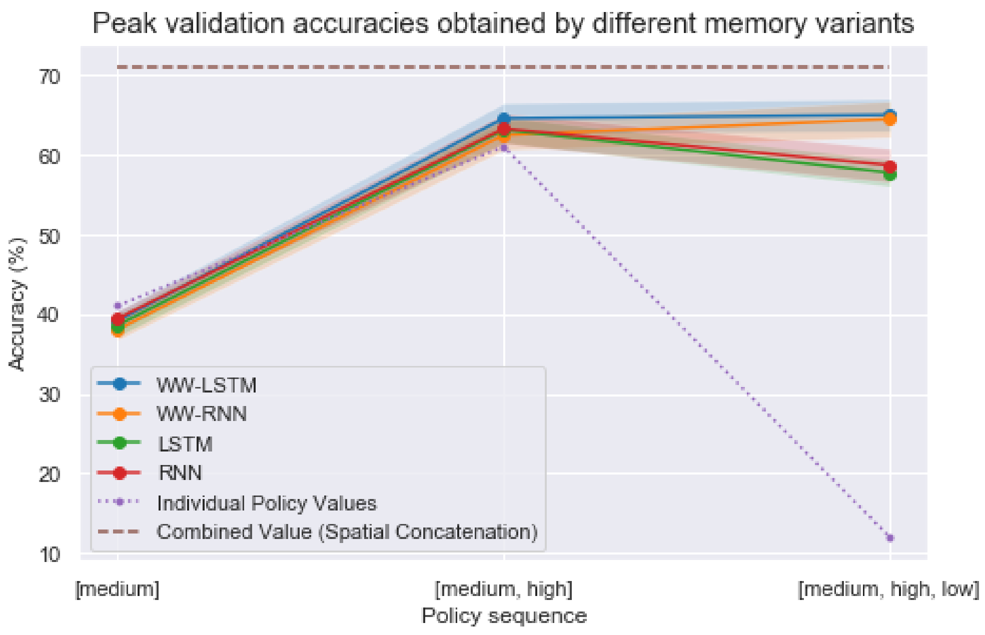


Figure 4.13: Best validation accuracies of modified recurrent attention models using different memory variants. Each data point represents the best result achieved during a training run with the model trained on a different policy sequence. Lines represent mean values, shaded regions represent the standard deviation.

The operation that constructs the WW matrix is strictly speaking an extension of feature extraction, although it compels a different structure for the LSTM as using fully connected layers at its gates would cause a large parameter count that would significantly complicate training. This warrants re-appraising the individual and combined hardcoded attention policies with an appraisal model that utilizes the WW operation; however, that is left for future work. It is also worth noting that no extensive hyperparameter search was conducted on the WW variants. The WW variants should be tested against un-pooled vanilla memory variants, as their low training losses raise the possibility of these variants' improved performance being a result of a larger hidden state tensor and a wider fully connected classifier layer.

## 4.7 Conclusion

This chapter presents active vision research that was conducted in a granular and tightly scoped manner, in line with the prescriptions laid out in Chapter 3. A novel method for investigating memory without relying on a learnable attention mechanism was laid out in Section 4.2; it argued that hardcoded attention policies can be appraised and used to control the value of each observation the system makes and that this can provide useful insight into memory's functional characteristics.

Section 4.3 implemented the aforementioned method by hardcoding, appraising and sequencing a number of attention policies for the Birds dataset. Section 4.4 used the sequenced policies to collectively evaluate them; it has revealed that concatenating the observations in input space performs really well as an aggregation strategy. The concept of temporal robustness has been introduced and evaluated in Section 4.5, filling in the final gap of the proposed method for investigating memory.

Section 4.6 evaluated a number of different recurrent memory variants, two of which consisted of a novel data structure - the WhereWhat matrix - and two operations specifically tailored to processing it. Memory variants utilizing the novel data structure performed significantly better than those that without it, but it wasn't enough to close the performance gap between recurrent memory and the spatial concatenation strategy.

This chapter presents an argument for using the COV method and for collectively appraising the attention policy sequence in active vision memory research. Temporal robustness was shown to be a relevant concern for memory that has no adequate solutions described in the literature. The results call into question a fundamental assumption that structured much of active vision research in deep learning; they do so by motivating the pursuit of approaches to active vision memory that are different to the Recurrent Attention Model architecture that was described in Section 3.2.2. The results suggest that an approach which shares functional characteristics with the spatial concatenation strategy might be preferable. A memory implementation different than the spatial concatenation strategy is highly desirable as it does not translate to more complex vision tasks, such as segmentation or object detection, in a straightforward manner. Based on the experiments conducted in this section the author believes that an effective memory implementation must preserve spatial information in its data format so as to facilitate further feature extraction.

# Chapter 5

# Attention

***Abstract:*** *The previous chapters of this thesis have established an active vision framework, investigated active vision memory and covered the software retina sensor. This chapter explores a number of questions focusing on the last remaining aspect of active vision: attention. Section 5.2 introduces a training curriculum for attention and demonstrates its benefits. Section 5.3 looks at the impact of the peripheral sensor image on classification and provides evidence suggesting that active vision systems might benefit from utilizing a two stream architecture. Section 5.4 proposes predictive attention, which is an alternative to using reinforcement learning for obtaining an attention policy. Finally, section 5.4.3 demonstrates the feasibility of using the hardcoded attention spotlight as a functional component of predictive attention.*

## 5.1 Motivation and Objectives

The overarching goal of this thesis is to advance the understanding of active vision in deep learning through developing and investigating an active vision architecture. To that end, this chapter explores the central problem of active vision: attention, or the means by which an active vision system decides where to look.

Active vision attention can be developed either as an integral part of the vision system's neural network architecture, or in the form of a stand-alone algorithm. The integrated approach is the one that was employed in this chapter. This was motivated by the fact that developing a stand-alone algorithm is the more challenging option which would likely require designing features by hand instead of leveraging deep learning. Additionally, active vision attention needs a form of memory in order to perform functions such as inhibition of return which, if implemented separately from the neural network vision components, would require duplicating much of the functionality already present in the system.

The experiments conducted in Section 3.3 have shown that reinforcement learning alone is not sufficient for enabling the active vision system to choose where to look in an effective manner. There is a breadth of reinforcement learning literature that could be employed in trying to improve on these results, while literature focused on integrated approaches that do not employ reinforcement learning is scarce. This chapter focuses on addressing this knowledge gap by investigating alternatives to reinforcement learning in developing active vision attention. The results obtained so far are sufficient in motivating the pursuit of novel approaches for training active vision attention, and this thesis has a better chance at producing a novel contribution by addressing a fundamental knowledge gap rather than by trying to integrate and improve upon the already mature and well-studied body of reinforcement learning literature.

The RAM architecture has been previously described in the literature review in Section 3.2.2 as the most mature and most commonly used architecture for active vision in deep learning, and it has been used extensively throughout this thesis. However, the results in Sections 4.4.3 and 4.6.3 showed that a simple spatial concatenation in the input space is a superior approach to memory in terms of classification performance. This is strong evidence in favour of supplanting the RAM architecture with a system that uses a different memory mechanism altogether. As developing and investigating a novel memory mechanism is left for future work, the specifics of its implementation are not known at this point in time. This complicates researching active vision attention, as the specifics of the attention mechanism's implementation are dependant on the model's architecture. In addition, researching and developing complete attention mechanisms using the RAM architecture as a platform risks producing results with limited generalizability.

This chapter addresses these concerns in several ways. Firstly, it explores research questions that are to a significant degree invariant to the implementation details of the attention mechanism. Secondly, it performs experiments using two models that utilize significantly different memory mechanisms. Lastly, it proposes a novel attention mechanism and investigates one of its aspects independently of any active vision model or memory mechanism.

## 5.2 Curriculum Learning

### 5.2.1 Motivation and Objectives

Curriculum learning for memory has been previously investigated by Li et. al. [12]. Although the results in Section 4.5.3 have found that curriculum learning for memory hinders performance in an active vision setting, the question of curriculum learning for attention remains unaddressed. As curriculum learning is a data-centric approach that is invariant to the internal workings of the model, it is an appealing subject of study for producing results that can generalize to future active vision architectures.

The objective of the experiments conducted in this section is to find out whether varying sensor parameters throughout training can improve an architecture's learned attention policy, as measured by its classification performance. The sensor will be parameterized to have a large field of view that gets progressively narrower throughout training. The motivating hypothesis is that a visual sensor with a larger field of view will make the attention task easier by being more forgiving of inaccurate fixations and by enabling the extraction of more visual information from the scene. The gradual scaling of the task's difficulty will allow the attention module to first learn simple policies and refine them as the curriculum progresses, resulting in improved performance relative to regular training.

It is worth noting that the exact opposite happens during human development: due to being unable to focus on distant objects infants start out with bad eyesight that progressively becomes better [100]. This might have a similar curriculum-like effect on their vision that initially simplifies the attention task by forcing the infant to focus on motion and large-scale features, and only later gradually increasing the amount of information they have to process.

### 5.2.2 Materials and Methodology

All experiments in this section were conducted using Pytorch 1.10.0+cu102 on a PC running Ubuntu 18.04 with a GeForce GTX 1080 Ti and an Intel Core i7-7700. The visual task is bird species classification on the Caltech-UCSD Birds 200-2011 dataset [61] that has been previously described in Section 3.3.2.

| **Stage** | **Epochs** | **Patch Size** | **Peripheral Patch Scaling Factor** |
|---|---|---|---|
| #1 | 1-25 | 148x148*px* | 2.5 |
| #2 | 26-50 | 111x111*px* | 3.333 |
| #3 | 51-75 | 74x74*px* | 5.0 |
| #4 | 76-100 | 37x37*px* | 10.0 |

Table 5.1: The details of each attention curriculum stage. Note that the size of the peripheral patch before rescaling is kept at a constant 370x370*px*, meaning that the scaling factor is indicative of the peripheral patch's acuity.

The experiments in this section compare models trained with and without an attention curriculum that comprises of modifying the models' sensor parameters. In all cases the models' attention modules are trained using reinforcement learning; full details about the reinforcement learning algorithm used can be found in the original RAM paper [10] as well as the DRAM paper [11]. Models trained without the attention curriculum utilize a sensor that extracts two overlapping patches out of the input image: a foveal patch of 37x37*px* and a peripheral patch consisting of a 370x370*px* image patch that has been downscaled to match the foveal patch's dimensions. Note that this is not the sensor configuration that has been used in the experiments in Chapter 4 as in that chapter the sensor consisted of only the singular, foveal patch. Models trained with the attention curriculum also utilize a sensor with two patches, with the foveal patch changing as the curriculum progresses and the peripheral patch always being a 370x370*px* patch that is rescaled down to match the dimensions of the foveal patch. Unlike the previously investigated memory curriculum which used the model weights from a previous training run as the initialization of the new model weights, the attention curriculum is executed during a single training run, with each of its four stages having a pre-defined duration ($N = 25$) in the number of training epochs. Table 5.1 shows the sensor parameters and the epochs during which they are applied. Note that the parameters at the final stage of the curriculum match those of the non-curriculum sensor. As the curriculum progresses, the field of view of the foveal patch narrows and the peripheral patch effective resolution decreases.

The two models used in these experiments consist of a modified RAM architecture with the WW-LSTM (defined in Section 4.6.2 as its memory mechanism and a model utilizing spatial concatenation (an expanded version of the spatial concatenation aggregation method defined in Section 4.4.2) as its memory mechanism. For simplicity these two models are referred to as WW-LSTM and SpaCat respectively. The use of two different models for this evaluation is motivated by the aim of producing results that are more generalizable to future active vision architectures, as previously discussed results have motivated the development of a system to supplant the RAM architecture. Both models were set to execute 5 time steps of interaction with the scene. The WW-LSTM model is almost identical to the architecture that has been previously described in Section 4.6.2 and was shown in Figure 4.9. The main difference is that here the output of the WW-LSTM's memory feeds into two additional modules: the attention module that determines the next fixation location and the baseline module used to regularize reinforcement learning. Additionally, the WW-LSTM model in this section combines the features extracted from the foveal and peripheral image patches by concatenating their respective WW matrices along the *'where'* dimension.

The SpaCat model's architecture can be seen in Figure 5.2. It is similar to the WW-LSTM model as it consists of the same ResNet18 feature extractor and broadly the same attention, classifier and baseline modules; however, this model aggregates the observations it makes in the image space, using spatial concatenation. The SpaCat model takes advantage of the matching dimensions of the peripheral and foveal patches by concatenating them together vertically to form a single observation image; the different observation images produced at each timestep are then concatenated horizontally. Figure 5.1 shows the structure of the input to the SpaCat model. The output of the SpaCat model's feature extractor passes through a WW operation before being used as input to the classifier, attention and baseline modules.

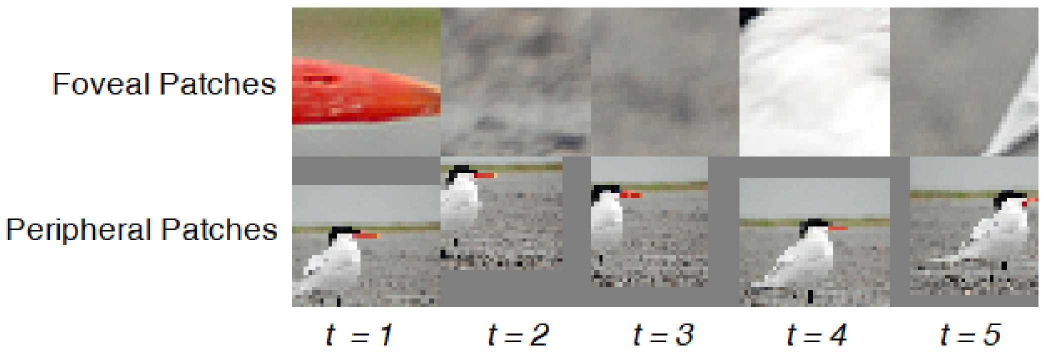


Figure 5.1: A labelled visualisation of the input format of the SpaCat model at the fifth and final time step, without an attention curriculum. The uniformly gray regions represent zero padding.

Unlike with recurrent memory mechanisms, the SpaCat model's memory representation does not maintain constant dimensions across time: as the model makes more observations it concatenates more observation images to its input, which in turn grows the size of the feature map produced by the feature extractor. This is problematic for two reasons, with the first one being that during long interaction sequences the input might consume an excessive amount of memory. The other problem is that if no global pooling is used then the size of the feature extractor's output will vary with the model's time step, preventing downstream modules from using its output directly as conventional fully connected layers expect an input of a fixed size. In this section this problem is alleviated by padding the SpaCat model's input with zeros to always match the shape it would have by the final time step. Additionally, it was found that the WW operation causes training instabilities and exploding gradients when used in tandem with large feature maps; to address this the initialization for the WW operation's weights was changed to He initialization [101] and a ReLU activation function was applied to the operation's output.

In all cases the models were trained for 100 epochs, with no early stopping employed. The starting learning rate was set to $5.0e^{-3}$. The feature extractor used was ResNet18 without the final average pooling and fully connected layers. The optimizer used was SGD with momentum set to 0.9 and weight decay set to $5.0e^{-3}$. The learning rate scheduler was set to halve the learning rate if the validation accuracy did not improve by at least $1\%$ over 5 epochs; however, for curriculum learning the internal variables of the scheduler were reset at the beginning of each curriculum stage. The batch size was set to 16 and the RNG seeds used were 1, 9 and 9001. Aside from changing the sensor parameters and resetting the learning rate scheduler's internal state, the start of a new curriculum stage also replaced the WW operation weights with new ones; this was necessitated by the fact that the dimensions of the feature maps produced varied with the curriculum stages, so new weights with updated dimensions had to be initialized. The attention module consisted of two fully connected layers, with their input and output sizes being (1024, 512) and (512, 2) respectively. The

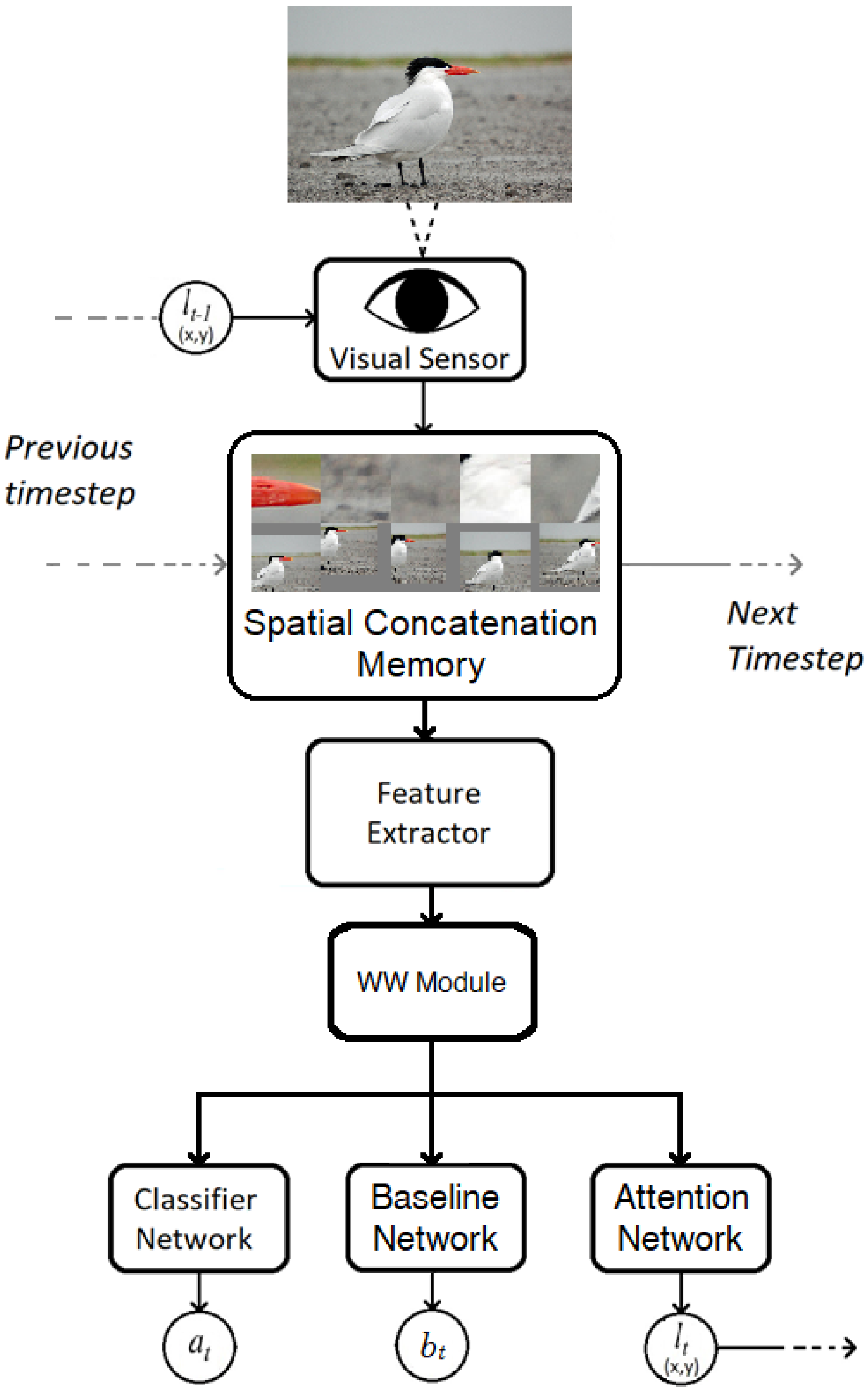


Figure 5.2: A diagram representing the SpaCat active vision model.

first layer utilized the ReLU activation function, and the output layer utilized the the $tanh$ activation function. The attention module came coupled with a baseline module that predicts the reinforcement learning reward it is going to get in order to reduce the variance of its loss.

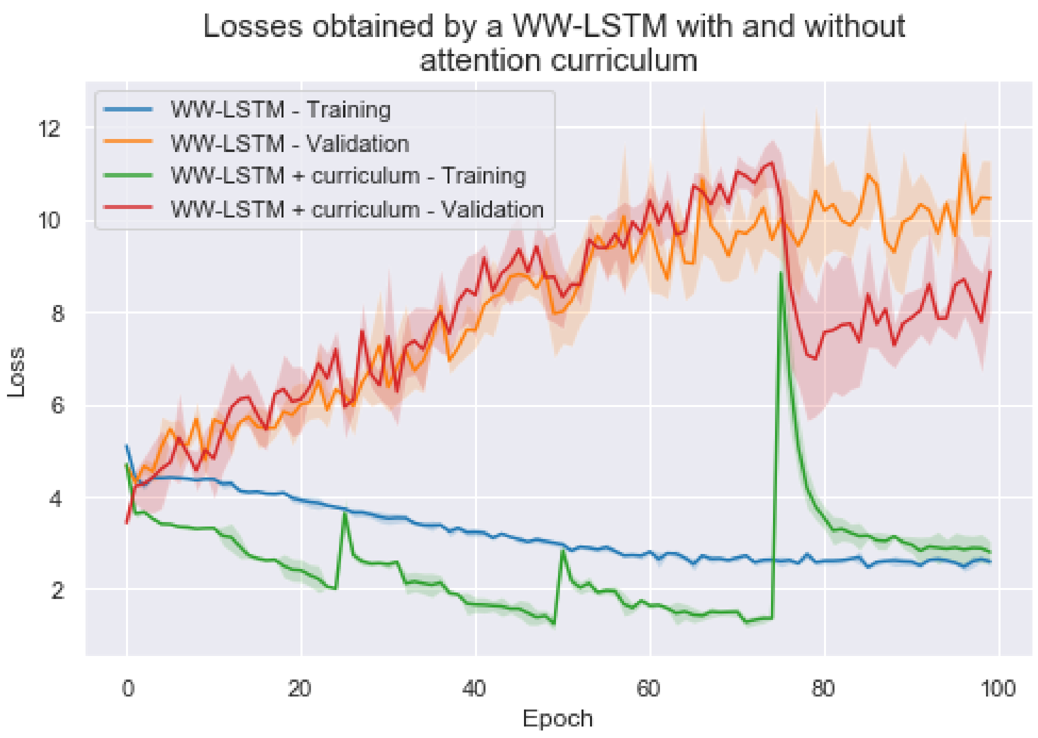


Figure 5.3: Losses of a RAM model utilizing WW-LSTM when trained with and without attention curriculum. Lines represent mean values, shaded regions represent the standard deviation.

### 5.2.3 Results and Discussion

The WW-LSTM losses shown in Figure 5.3 show that even though curriculum learning provided some benefits, it was not enough to redeem reinforcement learning by preventing the validation loss from rising throughout training. Interestingly enough, the validation loss for the curriculum variant has been rising in tandem with the non-curriculum variant up until the final stage of the curriculum, where it dropped significantly; one explanation for this could be that the sensor became too narrow to capture the features that the network was previously overfitting to, which is supported by the massive spike in the training loss that occurred at the same time. The validation and training accuracies of the curriculum-trained WW-LSTM model, shown in Figure 5.4, are much more correlated than the models' losses. Both validation and training accuracies fall at every stage of the curriculum, and the final validation accuracies of the curriculum and non-curriculum variants are effectively the same. These results suggest that the proposed attention curriculum has the potential to influence the WW-LSTM model's overfitting, but it does not appear to have an effect on its final performance in terms of classification accuracy at validation time.

The losses for the SpaCat model shown in Figure 5.5 show a slightly different effect to that seen in the loss plot of the WW-LSTM model. The validation loss can also be seen to decrease at the final stage of the curriculum, but not to the same extent. The SpaCat model

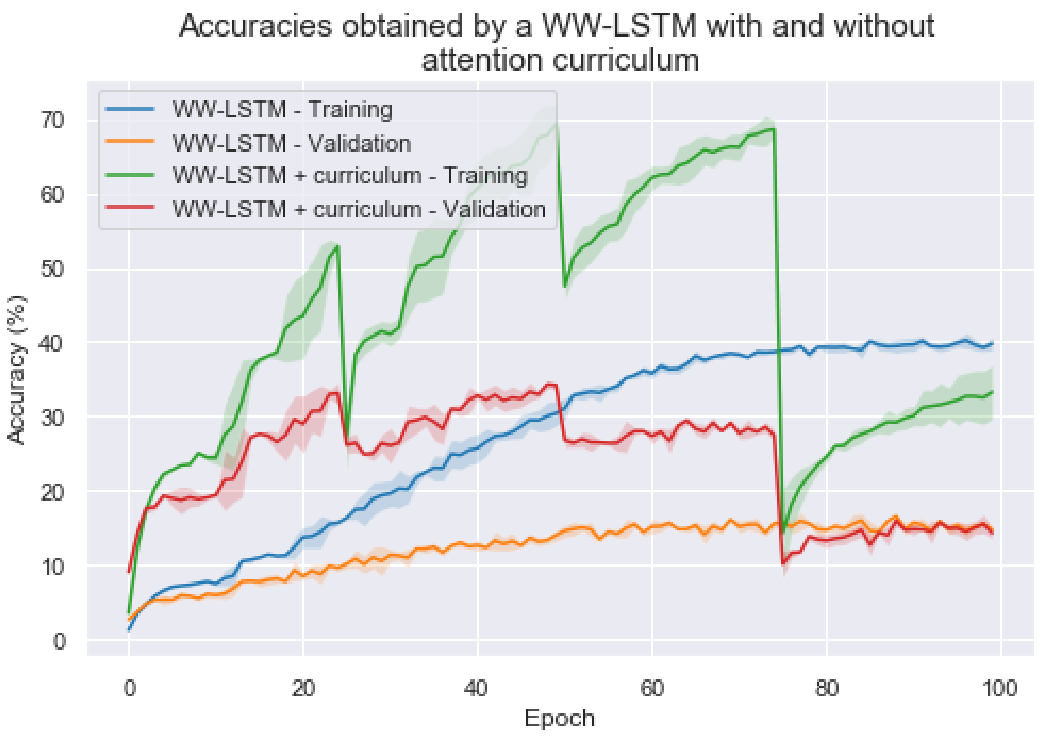


Figure 5.4: Accuracies of a RAM model utilizing WW-LSTM when trained with and without attention curriculum. Lines represent mean values, shaded regions represent the standard deviation.

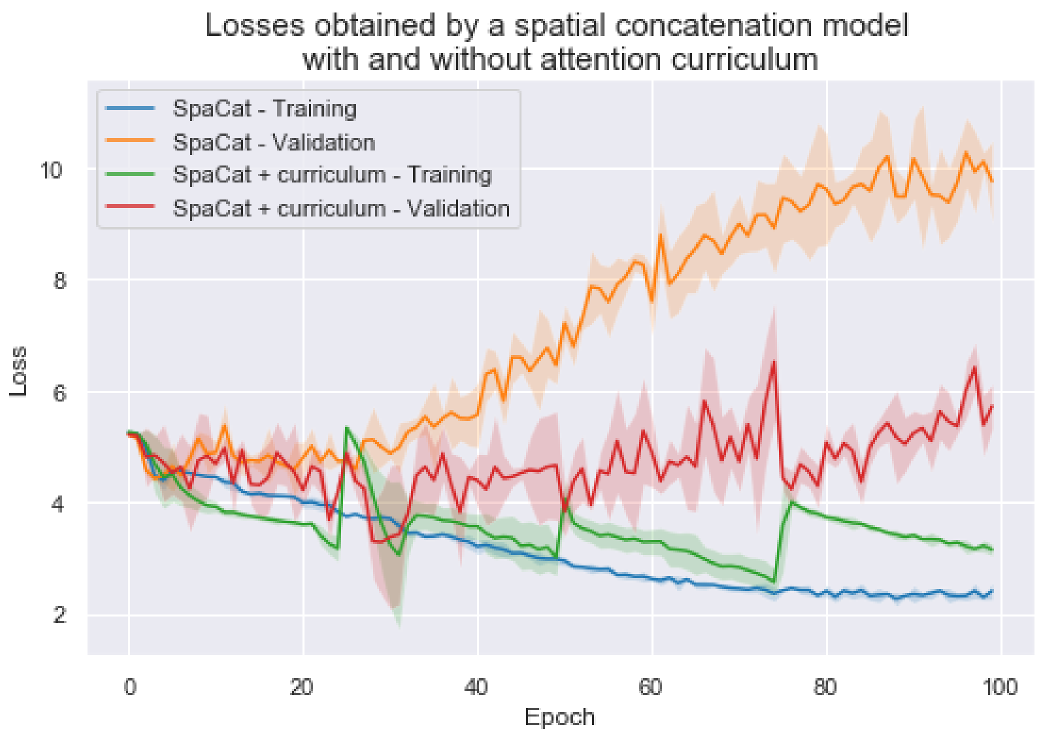


Figure 5.5: Losses of a spatial concatenation model when trained with and without attention curriculum. Lines represent mean values, shaded regions represent the standard deviation.

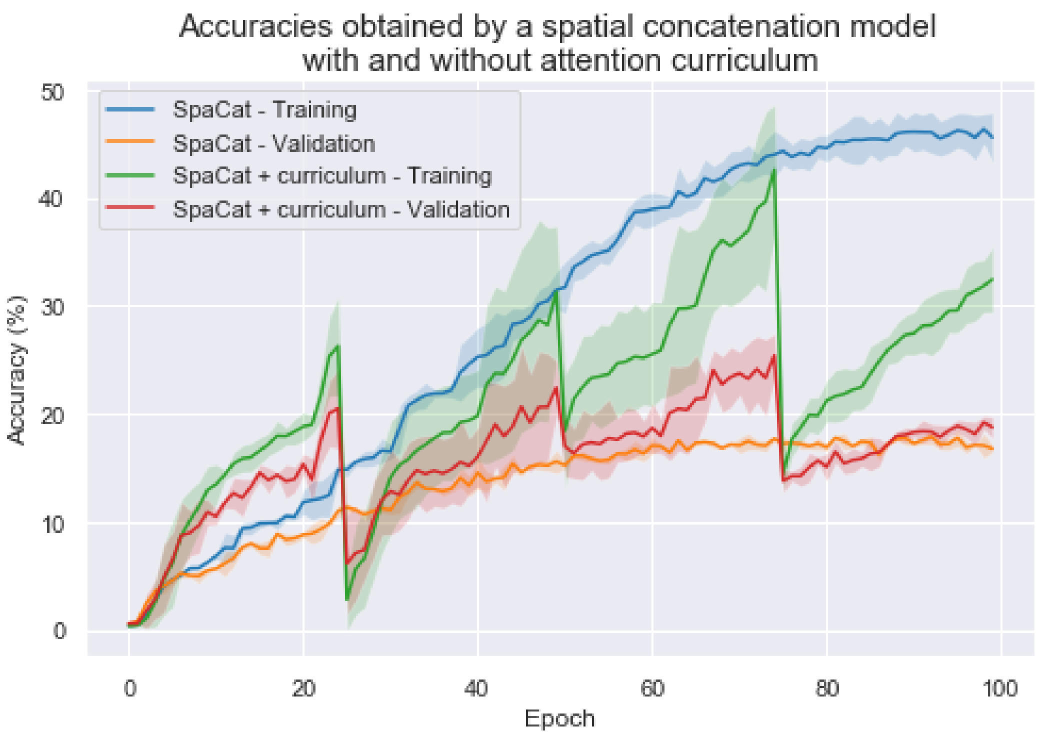


Figure 5.6: Accuracies of a spatial concatenation model when trained with and without attention curriculum. Lines represent mean values, shaded regions represent the standard deviation.

trained without curriculum learning has had its validation loss rise, but the curriculum variant's loss remained nearly constant between the start and the end of training. The proposed attention curriculum has reduced the SpaCat model's overfitting to a much greater extent than it has for the WW-LSTM model. The accuracy plot in Figure 5.6 shows that the attention curriculum variant's validation accuracy is slightly higher and that it has not fully settled by the end of training, suggesting that the model would have likely benefited from extending the final curriculum stage.

Overall these results motivate a further exploration of attention curriculum learning as the method has the potential to reduce overfitting across different active vision architectures with different memory mechanisms. There is nothing about these results suggesting that this effect might generalize to methods other than reinforcement learning, but at the same time any hypothetical reasons as to why it should not generalize are not immediately obvious. It is worth noting that many details of the curriculum implementation, such as utilizing cosine annealing or other cyclic learning rate schedules, were not explored and that there is significant potential for optimising the proposed approach. One hypothetical aspect of this experiment that could be confounding these results is that the raw volume of visual information used in training the feature extractor is much bigger for the curriculum-trained models. This might have led to better feature extractor performance, but it is not very likely as prior results in

Section 3.3.4 have shown that models with a larger field of view are more likely to overfit the training data; nonetheless, this could be addressed in future work by using a pre-trained feature extractor with frozen weights.

## 5.3 Foveal and Peripheral Vision

### 5.3.1 Motivation and Objectives

A foveated visual sensor presents itself as a potentially useful tool in developing an active vision system's attention mechanism, as its low resolution peripheral view enables evaluating a large area of the scene for potentially interesting regions using relatively little computational resource. However, the impact of the peripheral view on the system's classification performance is not clear. Prior results in Section 3.3.4 have shown that a sensor with a wider field of view can perform worse than one with a narrow field of view if it fails to fixate on valuable parts of the scene. At the same time there is the possibility that a low resolution peripheral view could promote overfitting to the training data as it may contain global image statistics that are sample-specific yet not informative of the bird species in the image. This raises the question of whether the peripheral patch can be detrimental or simply unnecessary to an active vision system's classification performance.

The objective of this section's experiments is to investigate the peripheral view's impact on classification performance. Although this experiment does not directly investigate the issue of learning an attention policy, it aims to inform an architectural decision that will have a significant impact on the development of active vision attention. In all two-patch experiments conducted thus far in this dissertation the foveal and peripheral image patches were processed in a single, unified stream that feed information to both the classifier and the attention mechanism. The experiments in this section will inform whether the foveal and peripheral data should be instead processed in two separate streams. As the results obtained in the previous chapter suggest that recurrent memory is not the optimal solution for active vision memory, the experiments in this section will be conducted using both the recurrent WW-LSTM architecture introduced in Section 4.6.2 and the SpaCat architecture introduced in Section 4.4.2.

### 5.3.2 Materials and Methodology

All experiments in this section were conducted using Pytorch 1.10.0+cu102 on a PC running Ubuntu 18.04 with a GeForce GTX 1080 Ti and an Intel Core i7-7700. The visual task is bird species classification on the Caltech-UCSD Birds 200-2011 dataset [61] that has been previously described in Section 3.3.2.

The experiments in this section compare models trained with two different sensor configurations: one using only the foveal patch and another configuration using the peripheral patch together with the foveal patch. The foveal patch dimensions are 37x37$px$ and it does not involve rescaling the cropped out image content, whereas the peripheral patch extracts a 370x370$px$ area of the image and downscales it ten-fold to match the foveal patch's dimensions. In order to isolate the influence of learnable attention on classification results and to produce results representative of using an effective attention policy, all models trained in this section utilize the $[medium, high, low]$ hardcoded attention policy sequence that was previously described in Section 4.3. As memory is not the focus of the evaluation in this section, the COV method previously described in Section 4.2 has not been employed.

The two models used in these experiments consist of the WW-LSTM and the SpaCat architectures previously described in Section 5.2.2. As the models utilize the $[medium, high, low]$ policy sequence they are set to execute 3 time steps of interaction with the scene and their attention and baseline modules are disabled. The RNG seeds used are 1, 9 and 9001. The maximum number of training epochs was set to a 100, with early stopping triggering if validation accuracy did not improve by at least 1$\%$ within 40 epochs. The batch size was set to 16, the optimizer used was SGD with momentum set to 0.9 and weight decay set to $5.0e^{-3}$. The initial learning rate was set to $5.0e^{-3}$ and a scheduler cut it in half every 10 epochs. All fixation locations had normal noise with $std = 4$ applied to them using a separate RNG seed.

## 5.3.3 Results and Discussion

Figures 5.7 and 5.8 show that all models have obtained comparable performance on the validation set aside from the WW-LSTM variant utilizing both the foveal and peripheral patches, which performed worse in terms of both validation accuracy and validation loss. That variant has also fitted the training set the fastest out of all the variants tested; its training loss is the first to approach 0 and its training accuracy is the first to approach 100$\%$. This suggests that the foveal+peripheral WW-LSTM variant has performed relatively poorly due to overfitting facilitated by the peripheral patch.

The invariance of the SpaCat model's performance to the presence of the peripheral patch suggests that spatial concatenation in the input space supports ignoring overfitting features present in the periphery; this is most likely facilitated by the peripheral patch always occupying the same location in the observation image, enabling the feature extractor to reliably learn to ignore its contents at the level of its convolutional weights. In contrast, the WW-LSTM model merges the foveal and peripheral data after feature extraction, leaving much

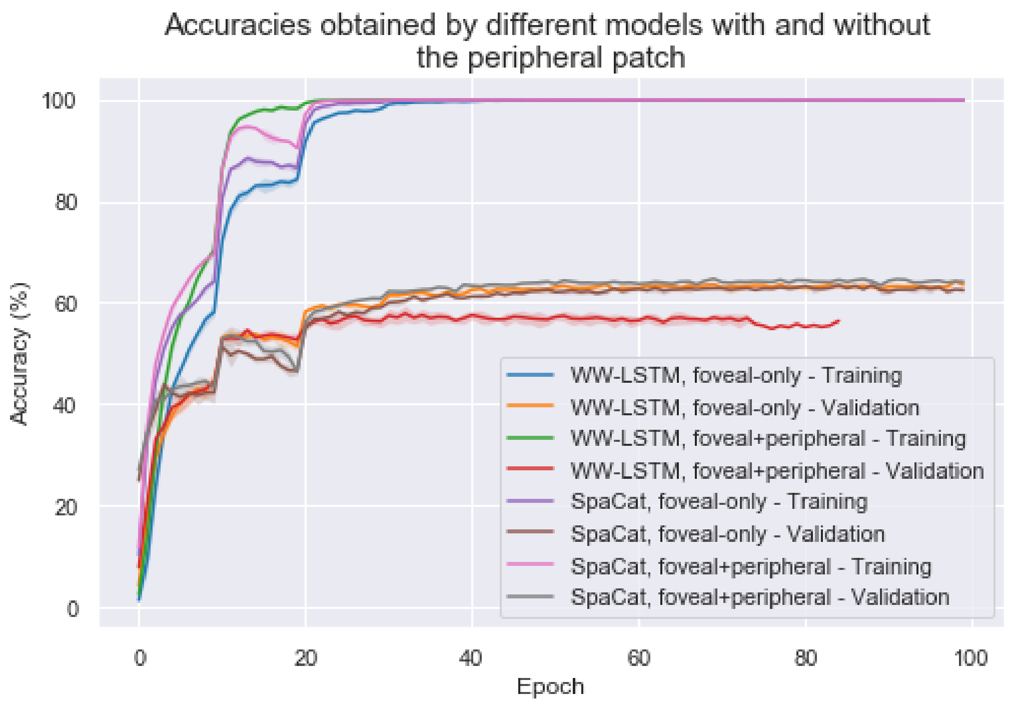


Figure 5.7: Accuracies obtained by the WW-LSTM and SpaCat models, with and without the peripheral sensor patch. Lines represent mean values, shaded regions represent the standard deviation.

less network depth to selectively suppress the peripheral data. This result suggests that if the peripheral and foveal images are to be fused then an early fusion is preferable; this is an insight that is similar to that found in Chapter 4 where spatial concatenation of multiple observations in the input space was found to outperform recurrent memory in the latent space. A broader insight might be drawn from this to say that these results support the leveraging of the feature extractor architecture in processing data present in disparate views of the scene.

The peripheral patch does not appear to have provided value to the classification task in any of the test cases, and has proven to be detrimental in one case. This result motivates the pursuit of active vision architectures that employ two separate processing streams: a foveal stream for classification and, as the hypothetical benefits to feeding the attention mechanism with foveal information are not clear, a peripheral stream for attention. This is in line with prior findings in physiological vision research in the work of Clayden et. al. [102]. Further questions are raised regarding memory and feature extraction, namely whether attention needs visual memory or whether a form of memory that consolidates proprioceptive information in the form of past fixation locations would suffice, and whether a shape-biased feature extractor would confer benefits over texture-biased CNNs [103], but investigating these is left for future work. The performance gap between the WW-LSTM and the SpaCat model that was previously observed in Chapter 4 is not present in the results for the foveal-only variants; this is most likely due to the SpaCat architecture being different from the spatial

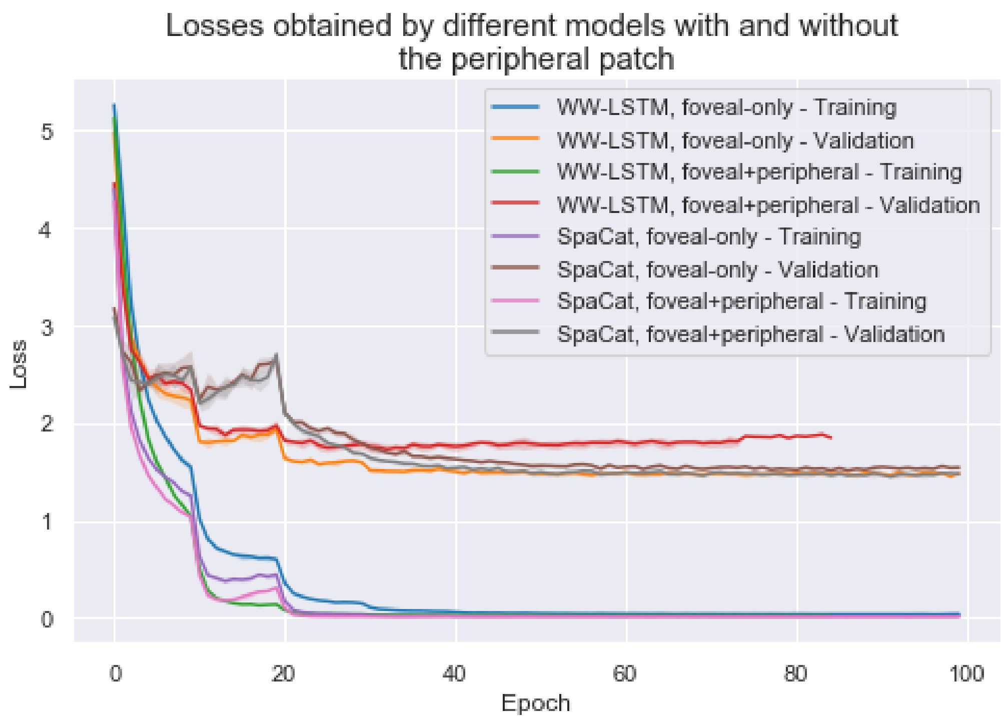


Figure 5.8: Losses obtained by the WW-LSTM and SpaCat models, with and without the peripheral sensor patch. Lines represent mean values, shaded regions represent the standard deviation.

concatenation architecture used in Section 4.4 as SpaCat's classification performance is notably lower.

## 5.4 Predictive Attention

### 5.4.1 Introduction

In a paper that predates the current deep learning boom by more than two decades Schmidhuber & Huber [83] describe an architecture for aligning an artificial fovea with a target object. The authors use a model network, i.e. a network that models the environment dynamics, as a substitute for backpropagating gradients through the environment in order to bypass the need for reinforcement learning. The model network takes as input the current environment state and the action to be taken by the architecture, meaning a view of the scene and the next fixation location, to predict the state of the environment after completing the action. The model network is trained in an unsupervised manner using random fixations. The difference between the model network's predicted view and the desired view is then used to compute gradients for training the controller network that guides the artificial fovea to the desired location at test time.

Although the work of Schmidhuber & Huber [83] only solves a toy active vision problem, the model-based approach presents itself as an opportunity for training active vision attention without resorting to reinforcement learning. Other model-based approaches have seen significant use in reinforcement learning for robotic control, yet there have been no investigations into utilizing such an approach for learning an active vision policy in deep learning. The most closely related approach within active vision literature is that of Jayaraman et. al. [13] who repurposed the model network, dubbed in their approach as the predictive module, to provide an additional, unsupervised source of gradients for training a part of their architecture.

Motivated by the poor results produced by reinforcement learning in Section 3.3, this section proposes the pursuit of learning an active vision attention policy using a mechanism for modelling the visual scene. Within this thesis, this approach is referred to as predictive attention. The goal of proposing predictive attention is to provide a research direction that guides future investigations towards an alternative to reinforcement learning.

### 5.4.2 Proposal and Discussion

The heart of predictive attention is the idea of using a modelling mechanism as a means of getting classification loss to produce gradients for training an attention policy. The approach described in the work of Schmidhuber & Huber [83] cannot be directly transposed onto contemporary deep learning vision systems because the problem tackled by that approach was too simple; it did not involve classification as it only focused on aligning a virtual sensor, and it utilized a simple, synthetic dataset. In addition, their architecture did not incorporate a feature extractor, and therefore did not utilize any latent spaces that complicate the development of predictive attention.

Figure 5.9 shows a conceptual starting point for predictive attention and explains the core idea behind it; it is not a complete representation of the proposed approach, but rather a simplified example used in the way of opening a discussion. In this architecture sketch, as suggested by the results from Section 5.3.3, the classification stream only utilizes information extracted from the foveal patch. The peripheral patch meanwhile is passed to the attention mechanism and, alongside the fixation coordinates produced by the attention mechanism, to the modelling mechanism. The modelling mechanism has been pre-trained to predict the foveal view at the specified coordinates using the peripheral view. The predicted foveal view is passed to the classifier, the output of which is then compared against a target to compute predictive attention loss. This loss is used to backpropagate gradients through the modelling mechanism all the way back to the attention mechanism. In this formulation, during the predictive attention backward pass, the modelling mechanism's weights are frozen, i.e. they are not updated.

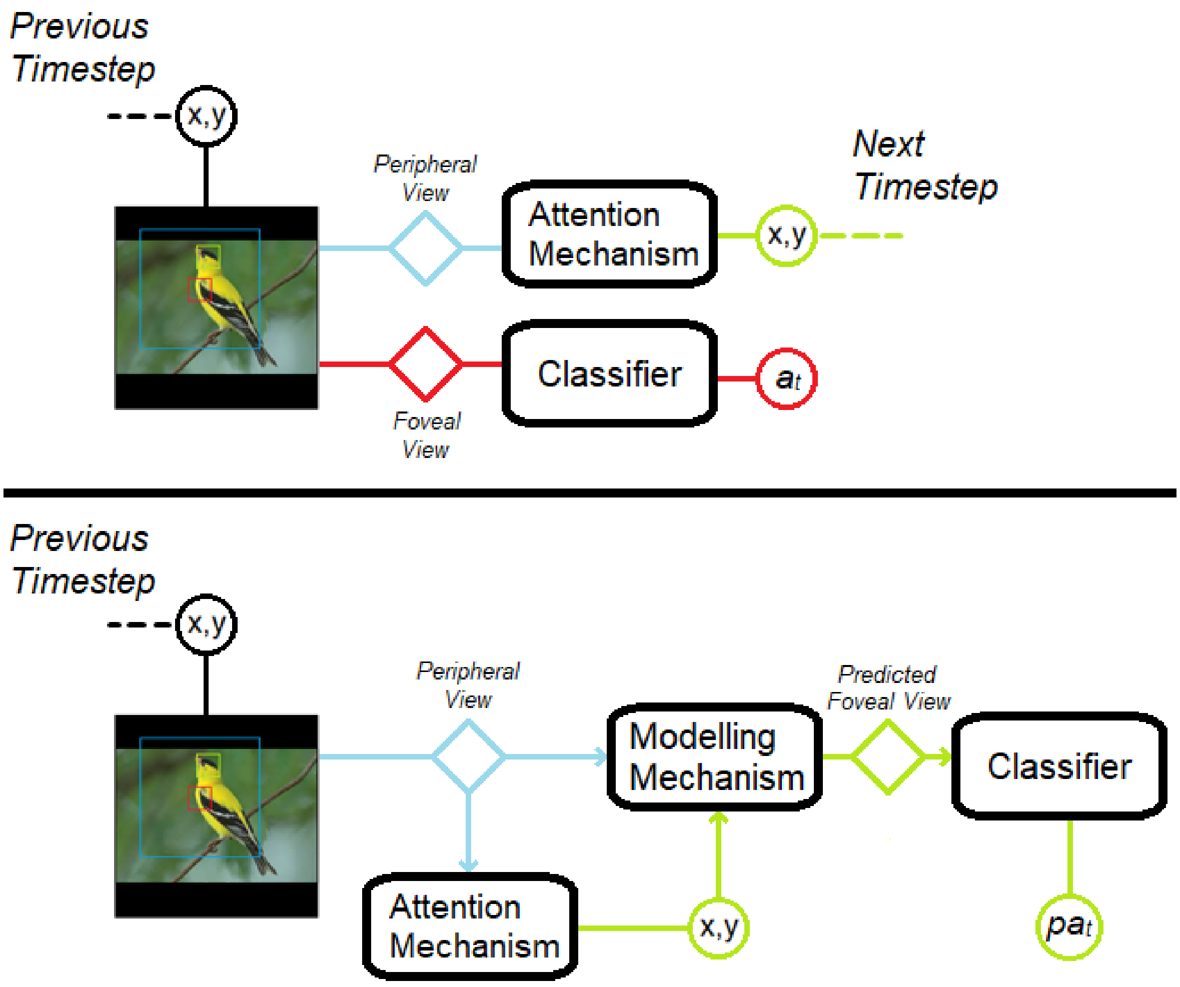


Figure 5.9: Diagrams showing a simplified overview of predictive attention. The sensor, feature extraction and memory have been omitted for simplicity. Red corresponds foveal view information, blue corresponds to peripheral view information, green corresponds to information pertaining to the next foveal view. $x, y$ represents fixation coordinates, $a_t$ represents classification output, while $pa_t$ represents a predictive attention classification output. **Top:** the organisation of the architecture at inference time. The peripheral view drives the attention mechanism to produce the next fixation location, while the foveal view drives classification. **Bottom:** the organisation of the architecture's predictive attention components at training time. The peripheral view drives both the attention and the modelling mechanisms. The predicted next foveal view, represented as the green diamond, is used as input to the classifier to produce a classification of what the stream predicts that it will see. This classification is instrumental in training the attention mechanism.

This simple example helps explain how predictive attention is intended to train an attention policy with classification loss. It also helps outline some of the issues that need to be addressed in the pursuit of predictive attention, the first one being the problem of the classifier's input domain. In the example shown in Figure 5.9, the output domain of the modelling mechanism is the same as the input domain of the classifier. This implies that the classifier has to have been trained before the modelling network's training commenced, which further implies that the classifier must have been trained without an attention policy as, after all, training an attention policy requires predictive attention. In other words, the different parts of this architecture are subject to circular dependencies during training. One hypothetical way to address this issue is to train this architecture in multiple phases, starting with an unsupervised bootstrapping phase that can train one of the necessary components independently of the others. Another way is to structure the backwards pass in such a way that all of the different parts of the architecture can be trained simultaneously.

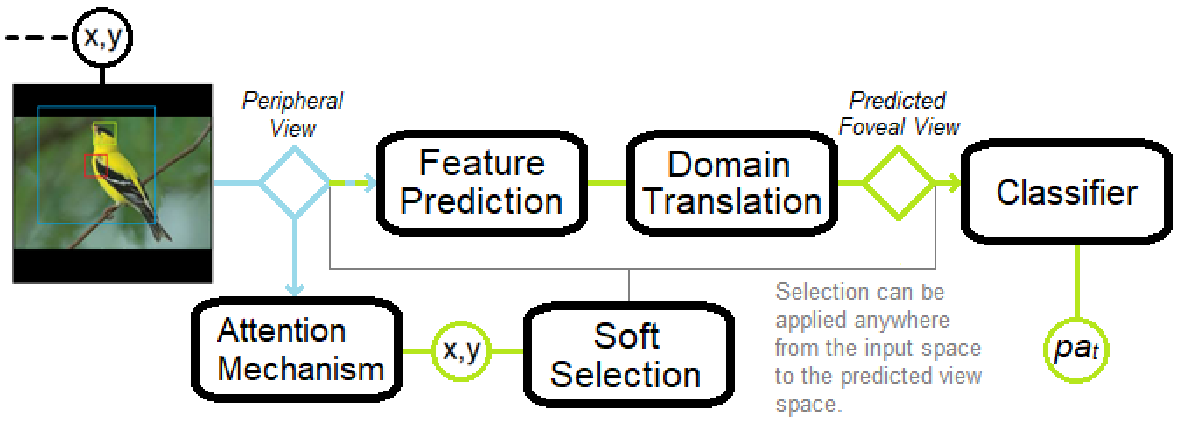


Figure 5.10: A diagram showing the decomposition of predictive attention into its functional parts: soft selection, feature prediction and domain translation. 'Soft' refers to the ability of gradients to be backpropagated through the soft selection mechanism. There is no immediately obvious theoretical constraint on where to apply soft selection within the decomposed approach. $x, y$ represents fixation coordinates, while $pa_t$ represents a predictive attention classification output.

The second issue related to predictive attention involves the modelling mechanism's forward pass, as it can be either structured in an end-to-end fashion or in a decomposed manner. Both the model network of Schmidhuber & Huber [83] and the predictive module of Jayaraman et. al. [13] are end-to-end approaches that utilize a fully connected layer which maps directly from an environment state and an action to a new environment state. The example shown at the bottom of Figure 5.9 also represents an end-to-end approach. The alternative is to split the modelling mechanism into multiple functional steps. These steps are: using the input coordinates to select an area in the peripheral view, predicting the content of the selected area as though it was viewed via the foveal region of the sensor, and representing that prediction in the latent space that is used as the classifier input. The first step of selecting an area of the peripheral view is elaborated upon in the next section (5.4.3). The second step, predicting the contents of the selected area as though they were viewed by the fovea, is closely related to image super resolution approaches as it predicts high frequency content from a low resolution representation. However, one significant difference is that the modelling mechanism is not expected to fully reconstruct all of the high resolution details but only those that are relevant to classification. The final step, of representing the predicted contents in the input domain of the classifier, can be either accomplished implicitly as a part of the previous step or explicitly as a latent space translation task. This decomposition of predictive attention is visualized in Figure 5.10.

The third and final issue highlighted by the above example is the fact that designing the backward pass of predictive attention, which details how classification gradients are used to train an attention mechanism, is an open ended problem that can be approached in a number of different ways. Three example approaches to its design are discussed alongside other future work in Section 6.2.2.

Listing the challenges inherent to predictive attention makes it clear that within the scope of this thesis it can only be pursued in a limited manner, as its development will be subject to several dependencies with the active vision architecture that it will be integrated with. The most notable dependency between predictive attention and the active vision system involves the input domain of the classifier, as the classifier always follows the memory mechanism which is subject to change since the results from Sections 4.4.3 and 4.6.3 have suggested the superiority of a simple spatial concatenation over the RAM architecture. The most feasible way for this thesis to contribute to the pursuit of predictive attention is to decompose the modelling mechanism's function and investigate one part of it in isolation. The following section investigates an approach for implementing the first functional step of the modelling mechanism: soft selection.

### 5.4.3 Hardcoded Attention Spotlight

#### Motivation and Objectives

The main objective of soft selection is to enable backpropagation from a latent feature space to x,y coordinate space so that the gradients which encode a desired change to the predicted foveal view can be translated into gradients that express a desired change in the input fixation coordinates. A simple 'hard crop' would not be acceptable, as it would completely mask out all regions of the peripheral view other than the region immediately surrounding the specified coordinates; if the information contained in those regions is not exposed to latter parts of the architecture then it would be impossible for backpropagation gradients to express a preference for information contained within those regions.

The proposed method for implementing soft selection for decomposed predictive attention is called the hardcoded attention spotlight, abbreviated to HAS. 'Hardcoded' refers to the fact that this method is not learnable. HAS can be thought of as an interface between the classification and attention streams in active vision, which is an essential component for developing decomposed predictive attention approaches. The role of the HAS algorithm is to produce an attention matrix to be multiplied with an input peripheral image or feature map. The attention matrix needs to fulfil a number of requirements in order to enable its use in predictive attention. It needs to:

1. Predominantly highlight information in the area around specified coordinates in the peripheral view. This is necessary to select the input information within the field of view that corresponds to the next fixation location.
2. Partially inhibit, but not completely eliminate information from the rest of the peripheral view.
3. Span the full peripheral view no matter the specified coordinates.
4. Facilitate appropriate backpropagation.
   (a) Be a form of 'soft attention', i.e. enable backpropagation into the specified coordinates.
   (b) Not be excessively flat; the matrix has to provide a gradient across the spatial dimensions of the input.
   (c) Not be discontinuous or excessively sharp. This is necessary to avoid getting stuck in a local minima.

### Algorithm Overview

The HAS algorithm presented in this section is a proof-of-concept prototype; it has not been tested within a predictive attention system. Listing 5.1 contains a code excerpt that shows how the HAS algorithm computes an attention matrix. The attention matrix consists of the *support* component that facilitates backpropagation overlaid with the *spotlight* which amplifies the information around the fixation point. The attention matrix contains values between 0 and 1, and is of the same spatial dimensions as the input image. What follows is first a brief overview and then a more detailed breakdown of the HAS algorithm.

To fulfil requirements 2,3 and 4 listed above, the HAS algorithm uses a Gaussian function as an infinite support in the attention matrix. This support Gaussian is necessary to maintain a smooth, spatial gradient at regions that are distant from the specified coordinates. The Gaussian function is differentiable, enabling backpropagation gradients to flow back into the coordinates that were used in its computation. To fulfil the first requirement listed above, constant values are added to the fixation area of the attention matrix. These values constitute the 'spotlight' of the attention matrix, as they serve to emphasize the area around the fixation location. The reason for these values being constants is that by not being derived from the coordinates, they are discarded during differentiation and do not contribute to backpropagation gradients.

Computation of the attention matrix by the HAS algorithm can be broken down into three major steps: computing a distance matrix, computing the support Gaussian, and finally adding the constant 'spotlight' values. Comments in the code shown in Listing 5.1 clarify which step is addressed by each block of code.

1. Given x,y coordinates and the desired size of the matrix, compute the distance matrix. Each pixel in the distance matrix contains its distance to the specified coordinates.
   (a) When using Pytorch: if the specified coordinates are to be optimised via back-propagation, instantiate them as an instance of *nn.Parameter*.
   (b) Create two matrices, *xd* containing its own x indices and *yd* with its own y indices.
   (c) Subtract the corresponding coordinate values from the *xd* and *yd* matrices.
   (d) Use the Pythagorean theorem to compute the distances of each matrix entry to the specified coordinates.
   (e) Clamp the distance matrix to prevent zero values.
2. Use the Gaussian function with the parameters specified in Listing 5.1 to compute the support component of the attention matrix.
3. Add the constant 'spotlight' values to the matrix. In Listing 5.1 the constants were chosen to approximate 1 when summed with the support values.

Using and computing the distance matrix using the method specified above is crucial in facilitating the backpropagation of gradients from an input image or a feature map and into a pair of coordinates. Listing 5.1 shows only one example parameterisation of the HAS attention matrix. It is worth noting that, in line with what is represented in Figure 5.10, the attention matrix produced by the HAS algorithm can be applied anywhere in the predictive attention stream, from the input pixel space up to the classifier's input.

Listing 5.1: A code excerpt showing how to produce the Hardcoded Attention Spotlight attention matrix at specified fixation coordinates.

```
import torch
import torch.nn as nn
import numpy as np



### STEP 1: Compute distance matrix
support_size = (9, 9) #size of the HAS matrix
```

```
#spotlight fixation coordinates
x, y = nn.Parameter(torch.tensor([4.0,4.0]))

B = torch.arange(support_size[0]).repeat(support_size[1],1)
xd = B - x
yd = B.T - y
distance_matrix = torch.sqrt((xd**2 + yd**2).clamp(min=1e-6))


### STEP 2: Use distance matrix to compute support Gaussian
def modGaussian(tensor, a, f, c):
      return a * torch.exp(-(tensor**f)/(2*c**2))

amplitude = 0.98 # amplitude of the spotlight
spotlight = modGaussian(distance_matrix, 1 - amplitude, 2, 10)


### STEP 3: Add constant values
def int_round(n):
   """Round to nearest integer and cast to int."""
   if n - np.floor(np.abs(n)) < 0.5:
      return int(np.floor(n))
   return int(np.ceil(n))

#Compute spotlight coordinates
width = 3 # pixel width of the spotlight area
from_y, to_y = int_round(y.item()) - width//2, \
            int_round(y.item()) + 1 + width//2
from_x, to_x = int_round(x.item()) - width//2, \
            int_round(x.item()) + 1 + width//2

#add spotlight to matrix
spotlight[from_y:to_y, from_x:to_x] += amplitude
```

### Materials and Methodology

To demonstrate the feasibility of the HAS algorithm an experiment was devised which aims to emulate its use with an input feature map when deployed as part of a predictive attention system. The experiment's objective is to test whether the spotlight can be guided to a desired location in the emulated feature map by iteratively updating its fixation coordinates using backpropagation.

A scene was initialized which serves to mimic a noisy feature map; it consists of a tensor with shape (37,37) filled with random numbers from a normal distribution with mean 0 and variance 1. A 3x3 area of the scene centered around $x, y = [1, 1]$ is designated to be the hotspot, with its values set to 40.

The HAS algorithm was set to produce attention matrices equal in size to the dimensions of the scene. The amplitude of the support Gaussian was set to 0.2 and its variance to 10. It was initialized at coordinates $x, y = [36.0, 36.0]$, which is at the opposite end of the scene to the hotspot. This coordinate set-up was chosen for this dissertation as forcing the spotlight to traverse diagonally across the entire scene is the most adversarial configuration possible.

An SGD optimizer was initialized with learning rate set to 0.06 and with the HAS coordinates as the only parameters for optimisation. The loss function was set to be the negative of the spatial sum of a multiplication of the scene with the HAS attention matrix, meaning that the optimizer was set to modify the HAS fixation coordinates in a way that maximizes the sum of the values captured by the attention map. The hypothesis is that thanks to SGD optimisation the HAS fixation coordinates would gradually shift towards the hotspot coordinates at $x, y = [1, 1]$ due to the hotspot's high values. The optimisation process was performed iteratively, with three stopping conditions: the number of iteration reaching 50 000, the loss being the same for 2 iterations in a row and the HAS coordinates reaching a distance equal to or less than 0.925 to the hotspot location. The first two stop conditions correspond to failures due to veering off course and getting stuck in a local minima respectively; the third stop condition corresponds to success.

In this experiment the constant 'spotlight' values were not added to the attention matrices. If they were used they would have been ignored during differentiation and would not influence the optimisation process; however, they would add excess noise to the resulting loss plots. The RNG seeds used were 1, 3, 6, 9, 919, 9001, 12345, 42, 1337 and 1984. The values logged at each iteration are the distance between the hotspot and HAS coordinates as well as the change in loss relative to the first loss value. The reason for logging change in loss instead of loss itself is that due to the stochastic nature of the experiment the starting and ending losses vary drastically between different RNG seeds, and logging change in loss allows for presenting results that are more regularized.

### Results and Discussion

In 7 out of 10 times the optimisation process has completed successfully, with the HAS coordinates approaching the hotspot location in under 30000 iterations in every case. Figure 5.13 visualizes progress snapshots of one such successful optimisation process. No run has

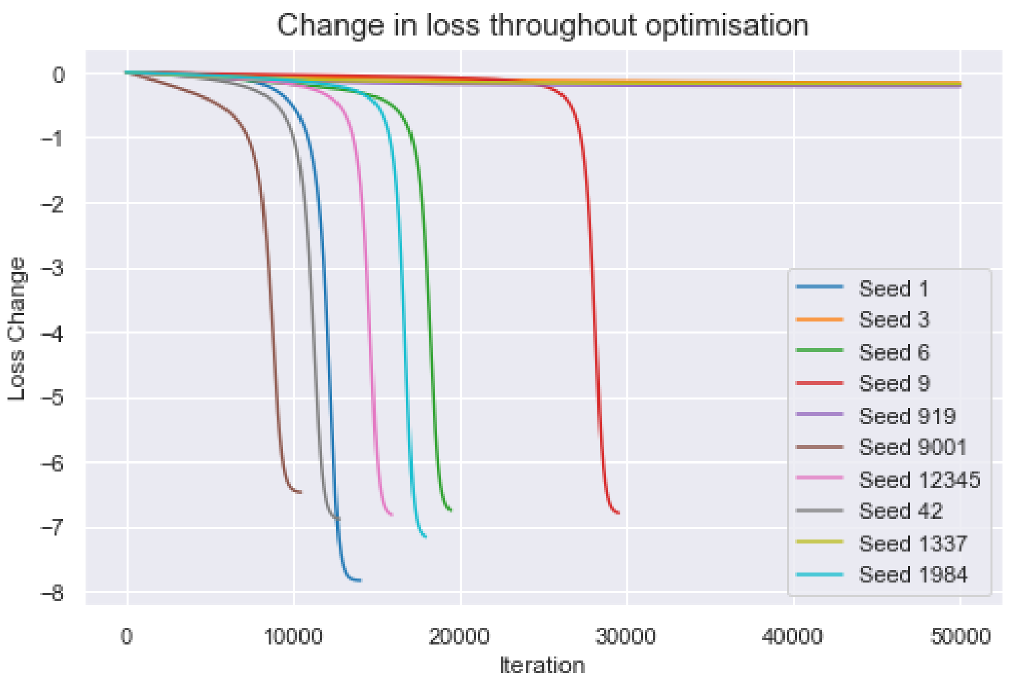


Figure 5.11: The losses obtained throughout optimising HAS fixation coordinates with different RNG seeds.

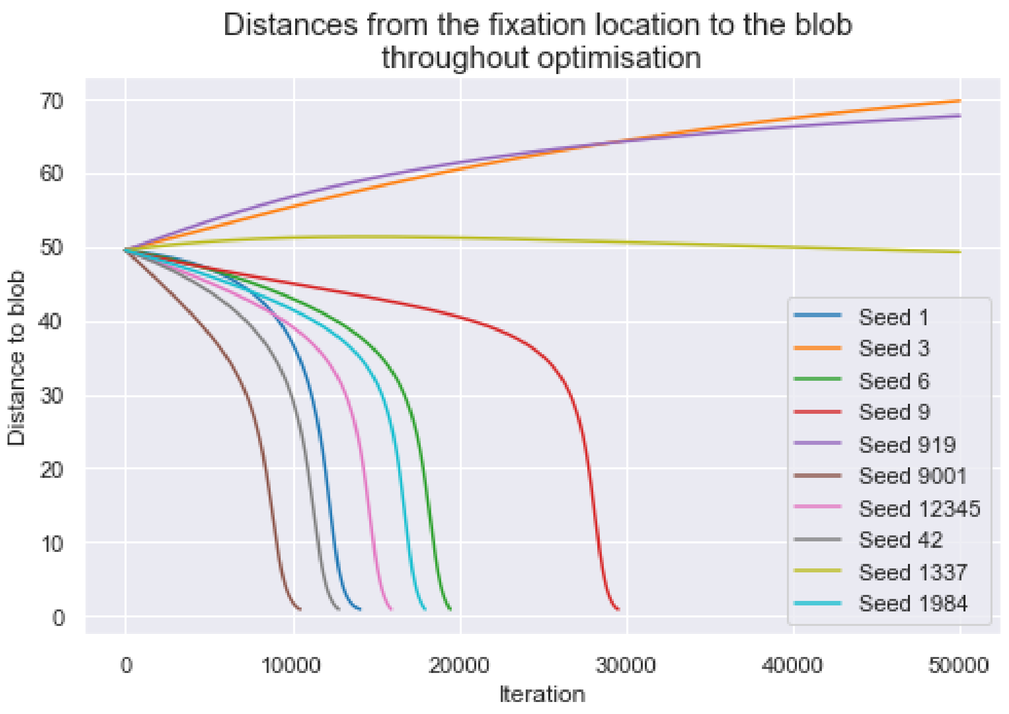


Figure 5.12: The distances between HAS and the hotspot throughout optimising HAS fixation coordinates with different RNG seeds.

been terminated due to HAS getting stuck in a local minima; however, during preliminary experiments a local minima relatively near - up to $5px$ away - the hotspot would occasionally result in HAS getting stuck. Increasing the standard deviation parameter eliminated this

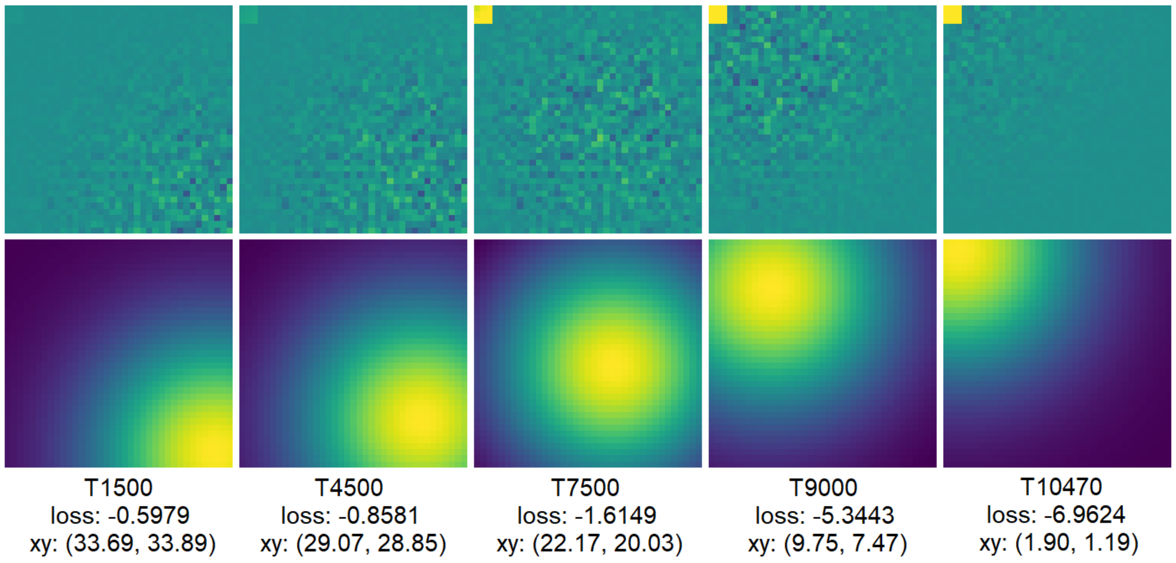


Figure 5.13: A visualisation of the optimisation process conducted in the experiment with RNG seed set to 9001. **Top row images:** products of multiplying the scene and the HAS attention matrices. **Bottom row images:** the HAS attention matrices. **Bottom text:** the timestep, loss and the HAS coordinates corresponding to the images in each column.

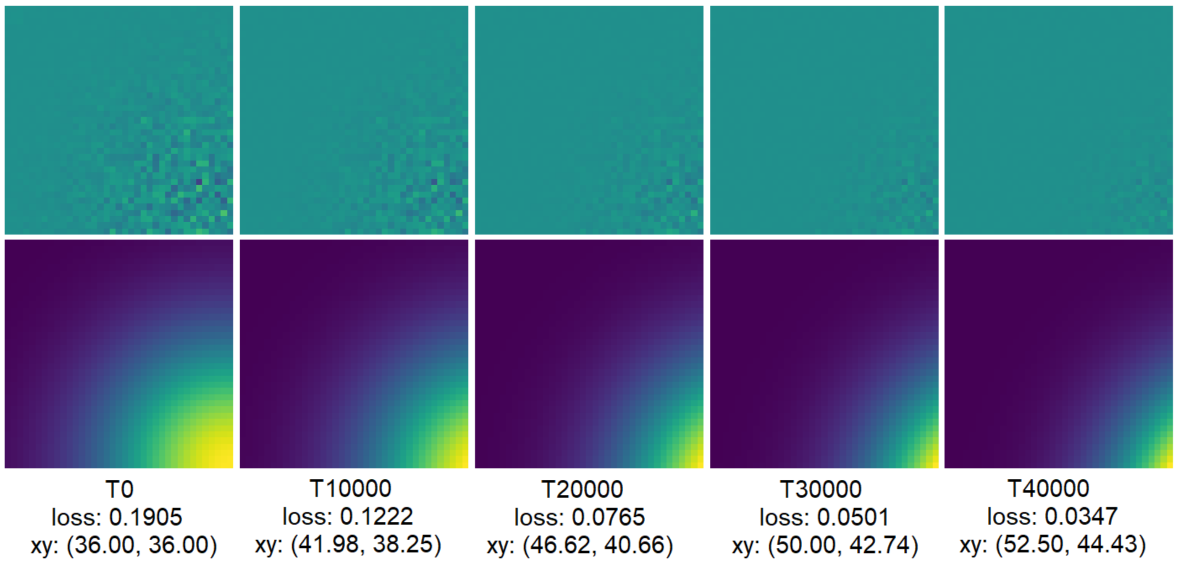


Figure 5.14: A visualisation of the optimisation process conducted in the experiment with RNG seed set to 3. In this run HAS failed to approach the hotspot and instead veered off the scene. **Top row images:** products of multiplying the scene and the HAS attention matrices. **Bottom row images:** the HAS attention matrices. **Bottom text:** the timestep, loss and the HAS coordinates corresponding to the images in each column.

occurrence.

In 3 out of 10 cases the optimisation process has veered off and has resulted in HAS escaping the coordinate bounds of the scene. Figure 5.14 visualizes progress snapshots of one such unsuccessful optimisation process; it shows how only a fragment of the support Gaussian is visible in the scene due to the spotlight being focused beyond the scene's boundaries.

This outcome was due to the local gradient around the initialisation point favouring moving away from the scene, which in turn was likely due to the presence of significant negative values between the starting HAS location and the center of the scene. The gradient signal from the negative values likely overpowered the gradient signal coming from the hotspot, leading backpropagation to guide HAS in the opposite direction to the one desired. This behaviour did not occur if the HAS coordinates were initialized closer to the center of the scene. Considering the adversarial configuration of the hotspot location and the initial HAS coordinates, this is a rare edge case that does not need to be addressed as it would not be likely to affect the function of a predictive attention system.

The results shown in this section have demonstrated the feasibility of using the HAS as a soft crop mechanism within a predictive attention system. The prototype works within the specified experimental conditions; however, in a predictive attention system the HAS attention matrix would be applied to input that contains numerous input channels, and its objective function would not be as simple as fixating on a hotspot area. Any hypothetical issues that could arise in such a setting are not immediately obvious, and thus investigating HAS in a more complex setting is left for future work.

## 5.5 Conclusion

This chapter presents active vision research investigating several approaches regarding the attention mechanism that enables an active vision system to choose where to look. Care was taken to conduct research that can generalize to future active vision architectures due to the findings in Chapter 4 questioning the use of the RAM architecture. Section 5.2 evaluated a training curriculum for attention and found it beneficial in reducing the active vision systems' overfitting.

Section 5.3 looked at the influence of peripheral vision on the system's classification performance. By demonstrating no benefit and occasional detriment of including peripheral vision in the classification stream, the results provide a motivation for the pursuit of two stream architectures for active vision: a peripheral attention stream and a foveal classification stream. The results in this section have also provided further evidence in support of fusing multiple observations prior to feature extraction.

Section 5.4 has proposed and discussed predictive attention, which is a method for training an attention policy designed to supplant reinforcement learning. The proposed method introduces a model of the environment that enables using classification gradients in training attention. Several possible approaches to structuring such a modelling mechanism were discussed. Section 5.4.3 investigated one functional aspect of predictive attention by evaluating the hardcoded attention spotlight, which is a method for soft cropping a region from the peripheral view in a way that enables gradients to be backpropagated into the crop coordinates and the attention mechanism that produced them.

This chapter concludes the investigations into active vision that were conducted as part of this thesis. Due to the numerous outstanding questions pertaining to active vision memory and the broader architecture of the active vision system, no complete solutions to attention were investigated. Instead, this chapter focused on producing knowledge that can support and guide future research into active vision attention.

# Chapter 6

# Conclusions

***Abstract:*** *This chapter summarizes the research conducted in this thesis by enumerating the main contributions to the literature. It closes with a recommendation of future work that can further advance our understanding of active vision in deep learning.*

## 6.1 Contributions

At the beginning of this thesis, in Chapter 1, the following five objectives were established to guide the research conducted in this Ph.D.:

1. To investigate the suitability of a retina-like biomimetic sensor for active vision systems.
2. To establish and follow a research practice that can address the knowledge gap present in the active vision literature.
3. To produce foundational knowledge that can guide future active vision research.
4. To investigate how does an active vision system collate the information that it obtains across multiple observations.
5. To investigate different approaches in enabling an active vision system to decide where to look.

The following sections summarize this thesis by outlining its major contributions and discussing how they address the five objectives listed above.

### 6.1.1 An Investigation and a Demotivation of the Software Retina

Chapter 2 describes the research efforts undertaken with respect to the software retina. It opens with an explanation of the initial objectives guiding its pursuit, which were to expand the retina's functional capabilities by selectively replicating the computations that take place in animal retinas and visual cortices, and to develop an adequate experimental environment for a rigorous evaluation of the software retina. The retina's functional capabilities were expanded with a reproduction of the retinal scale space pyramid previously described by Balasuriya [3] and a retinal colour opponency model that was described in Ozimek et. al. [32]. An experimental environment for evaluating the software retina has been devised, using a subset of the EPIC Kitchens dataset [40] and a modified Recurrent Attention Model [10] architecture.

In order to validate the function of the modified RAM architecture, it was first evaluated on the EPIC Kitchens egocentric activity recognition task without using the retina. The architecture demonstrated meaningful learning and performed favourably compared to a benchmark; however, when the software retina was integrated with the architecture it slowed the training process down to an unacceptable pace. This was caused by a number of factors, including the computational overheads introduced by the retina, its inefficient integration with the deep learning ecosystem and the technical challenges associated with processing a video dataset in an active vision setting. As rapid architectural revisions and evaluations were deemed necessary to conduct an adequate volume of research, a decision was made to shift the focus of the thesis away from the retina and to instead focus on active vision.

The software retina has been re-visited in one experiment in Section 3.3, where it was integrated with an active vision architecture using the cortical mapping [1]. Unlike in the EPIC Kitchens setting, the computational load of this experimental setting has been sufficiently low to allow a simple evaluation. The retina was evaluated on an image classification task on the Birds dataset and has been shown to be inferior as an active vision sensor when compared to cropping out square patches from the input image; it performed significantly worse in terms of classification accuracy and it increased training time by a factor of 7.56.

The research described above has addressed the first objective set out in this thesis by demonstrating that the software retina is currently not suitable for active vision systems.

### 6.1.2 A Demonstration of Knowledge Gaps and a Framework Proposal

A change in the focus of the thesis warranted a re-formulation of its goals and motivations, as well as a new literature survey. Chapter 3 contains a comprehensive commentary on the active vision in deep learning literature and its shortcomings, together with experiments that exemplify its numerous knowledge gaps. The review of the literature has found it difficult to browse; some publications suffer from a lack of adequate referencing and most researchers failed to use accurate terminology to describe their work. The inconsistent terminology made the review especially challenging, as a significant proportion of the relevant publications did not even refer to their own work with the term “active vision“. Meanwhile, many researchers who did refer to their work as “active vision“ have at times ended up tackling entirely different problems. An over-reliance on utilizing simple approaches to tackling research problems and a scarcity of publications that meaningfully contribute to prior active vision literature were identified as the primary causes for numerous knowledge gaps. Finally, the review uncovered that a lot of the work described in the literature simplified the active vision setting to the point of trivializing the associated problems instead of advancing our understanding of how to solve them.

The experiments accompanying the literature review were conducted with the goal of demonstrating the knowledge gaps in the literature in a practical manner. The Birds dataset [61] was selected as the most suitable dataset for the investigations due to the fine-grained nature of the bird species classification task and the utility of the dataset’s anatomical part location labels. Once again, the Recurrent Attention Model was used as the active vision architecture of choice. The research questions were deliberately selected to highlight gaps in how different methods were described in the literature. The results turned out to be critical of the literature, as the findings pointed out fundamental issues that ought to already have been explored in the literature. It was found that proprioception is detrimental to the system’s classification performance, that absolute image coordinates cause the system to overfit the training data, that reinforcement learning is inadequate for learning an attention policy and that visual sensors with larger fields of view can alleviate but not completely compensate for a poor attention policy.

The literature review and the experiments that demonstrated its gaps culminated in the proposal of an investigative framework for structuring active vision research. The framework breaks down passive vision systems into two components: the feature extractor and the mechanism for using the extracted features to complete the system’s visual task. The framework then defines an active vision system as complementing these two components with three additional architectural requirements: a sampling structure that defines the active agent’s local sampling strategy, an attention mechanism which implements the agent’s global sampling strategy, and a memory mechanism that implements the agent’s strategy for aggregating visual data collected across multiple observations. All research that followed Chapter 3 has been guided by this framework.

The research described above addressed the second objective set out in this thesis by proposing an investigative framework for researching active vision and by making sure that this framework is informed by the shortcomings present in the existing literature. Arguably, it also contributed towards the third objective by guiding future active vision research.

### 6.1.3 Active Vision Memory Investigations

Following the recommendations laid out in the proposed active vision framework, Chapter 4 investigated active vision memory. It opened with a discussion of the problem of investigating an individual aspect of active vision in isolation, without having complete solutions to other components of active vision. The lack of a suitable learnable attention mechanism presented itself as a challenge, as the system's ability to make high quality observations was seen as a necessity for testing the function of memory in the high performance regime. In response, an experimental method was devised, which involves controlling observation values to produce interpretable results about the function of active vision memory.

Applying the method for controlling observation values required the selection and appraisal of hardcoded attention policies that consisted of sequenced anatomical part locations. The part locations were provided as part of the Birds dataset. The concept of temporal robustness was established and investigated using several different approaches. A number of feedforward and recurrent memory mechanisms were investigated, and a novel recurrent memory mechanism called the WW-LSTM has demonstrated the best classification performance at a favourable parameter cost when compared to other recurrent variants. However, it was also discovered that a simple spatial concatenation of the different observation images in the input space was sufficient to outperform all recurrent memory variants. This result put into question one of the core ideas behind the RAM architecture, which is placing a recurrent memory mechanism after feature extraction.

The research described above addressed the second objective set out in this thesis by devising an experimental method for evaluating active vision memory without the need for a learnable attention mechanism. It has also addressed the fourth research objective by applying the aforementioned method and investigating different approaches to active vision memory, most notably the WW-LSTM and the spatial concatenation method.

### 6.1.4 Informing Future Active Vision System Design

Producing knowledge that can generalize to other research settings and that can inform future active vision investigations has been a priority throughout this thesis. As a result, the contributions that support this objective are scattered throughout different chapters.

The experiments conducted in Chapter 3 provided evidence supporting various different active vision design choices. An egocentric coordinate frame was found to be preferable to an exocentric one as it drastically reduces overfitting to the point of improving validation performance. Proprioception was found to be detrimental to the system's performance when using hardcoded attention policies and when training attention with reinforcement learning. Simple sensors consisting of overlapping image patches were found to be superior to the software retina in terms of their impact on classification performance and training speed.

Chapter 4 informed future active vision system design by providing evidence suggesting that a memory mechanism located within or prior to the feature extractor has the potential to outperform the RAM architecture, which at the time of writing this thesis is the staple active vision architecture. Additionally, curriculum learning for memory was shown to have a significant negative impact on model's performance. Although curriculum learning showed negative results for memory, in Chapter 5 an attention curriculum has demonstrated a reduction in overfitting. The attention curriculum starts with a sensor that has a large foveal sensor patch and a relatively sharp peripheral patch, only to gradually reduce the foveal field of view and the peripheral acuity throughout training.

The research described above addressed the third objective set out in this thesis by producing empirically backed arguments in favour of various methods for active vision, such as: utilizing a two-stream architecture that separates classification and attention, collating multiple observations in a manner similar to input-space spatial concatenation, and utilizing an attention curriculum that varies sensor parameters throughout training.

### 6.1.5 Reinforcement Learning, Predictive Attention and the HAS algorithm

Section 3.3 found that models using reinforcement learning to learn where to look obtained only a fraction of the validation performance that models with hardcoded attention policies reached. At the same time, the literature survey in Section 3.2 found that most active vision research utilized reinforcement learning for attention and did not pursue an alternative approach.

The aim of Section 5.4 was to respond to the uncovered inadequacy of reinforcement learning. This section proposed and took the first step towards developing an alternative approach for training attention in an active vision system. The proposed alternative is called predictive attention, and it aims to leverage classification gradients in training attention. The first step in its pursuit was an investigation into the hardcoded attention spotlight (HAS) algorithm, which is a prototype algorithm enabling the backpropagation of gradients from attention-modulated feature maps into fixation coordinates.

To investigate the HAS algorithm, a scene was instantiated that models a noisy feature map. A hotspot was placed in a corner of the feature map, and an optimisation process was established which aimed to guide the HAS fixation location onto the hotspot by maximizing the sum of the values captured by the HAS attention matrix. The role of this experiment was to provide a proof of concept for the HAS's suitability for deployment in a predictive attention system, where it would be used to translate classification gradients into updates to the system's attention policy. The results were positive, with only a fraction of test runs in an adversarial edge case failing to optimize the fixation location onto the hotspot.

The research described above addressed the fifth objective set out in this thesis by demonstrating the inadequacy of reinforcement learning for training an attention policy and by taking the first step in the pursuit of a novel approached aimed at supplanting reinforcement learning in active vision.

## 6.2 Future Work

This section proposes several investigations that, if undertaken, would meaningfully build upon the contributions presented in this thesis. As previously argued in the proposed research framework, the author of this thesis believes that mature solutions to active vision attention and memory are necessary for a rigorous investigation of active vision sensors; therefore, no future work focusing on sensors is proposed in this section.

### 6.2.1 Memory

#### Spatial Concatenation-Inspired Memory

The undermining of the Recurrent Attention Model's (RAM) position as the architecture of choice for active vision was an unexpected result. Although the proposed research framework has argued in favour of conducting tightly scoped research that either focuses on individual aspects of active vision or their interrelations, the development of a complete architecture that can supplant RAM should be a long term field-wide objective. Memory is central to active vision architectures, and is thus a critical component of developing novel, complete approaches. Spatial concatenation in the input space has demonstrated desirable performance as a memory mechanism, but it also exhibits undesirable characteristics such as dynamic memory tensor size and a relatively high memory footprint. Devising an architecture capable of leveraging the desirable aspects of spatial concatenation while overcoming its shortcomings would be the immediate next research objective for the author of this thesis if this line of research were to continue. A promising starting point for the pursuit of such a system would be investigating the potential of the NARX architecture [95] to be used as a recurrent memory mechanism that operates within the input space of the active vision system.

#### Residual Recurrent Memory

It has been previously argued in the literature that the ResNet architecture is equivalent to a special case of a recurrent network [104]. This argument can motivate the pursuit of a memory mechanism that is closely integrated with the active vision system's feature extractor. Such a solution would represent a compromise between RAM's approach to locating memory in the latent space and aggregating it in the input space as is done in spatial concatenation. One way of achieving this would be to insert recurrent memory modules, such as the convolutional LSTM [46], at the skip connections in residual learning architectures.

#### Further Study of Temporal Robustness

Section 4.5 has introduced the concept of temporal robustness and argued in favour of its relevance for active vision systems. Further investigations into this concept are necessary in order to enable the design of flexible active vision systems that can operate within dynamic computational time. Such investigations should be conducted alongside the development of novel memory mechanisms, and should attempt to reproduce the method deployed by Li et. al. [12] that utilized a separate classifier for each time step.

### 6.2.2 Attention

#### Dorsal Stream: a Shape Biased Feature Extractor

Dorsal stream, also known as the "where" pathway, is the neural pathway that is involved in spatio-visual reasoning [98]. The results of experiments conducted in Section 5.3 motivate the design of a separate stream for processing attention in active vision. The study of such a stream could benefit from drawing inspiration from the organisation and function of the dorsal stream in animal vision systems. A key question that ought to be investigated is whether the feature extraction architectures that excel at image classification are suitable for attention. Literature has shown that contemporary CNNs are biased towards texture and that additional methods need to be employed to bias them towards shape [103]; this raises the possibility of developing a feature extractor with an inductive bias towards shape and evaluating its impact on active vision attention. Other questions to explore include augmenting attention with the WW operations that were described in Section 4.6.2, as well as investigating the role of memory and proprioception in attention.

### QKV Active Vision Attention

Vaswani et. al. [105] popularized the Query-Key-Value (QKV) attention module in deep learning by introducing the Transformer architecture. This module has not been evaluated in an active vision setting; however, Mott et. al. [106] have used it to implement soft, spatial attention in a recurrent setting. Their approach demonstrates the feasibility of embedding QKV attention in a recurrent architecture, which could motivate the use of QKV attention in active vision attention. The structure of this module has the hypothetical potential to result in a more interpretable form of active vision attention, as the query tensor could be understood as an expression of what the model is looking for at a given time step.

### Further Study of Attention Curriculum Learning

The attention curriculum described in Section 5.2 demonstrated promising results, but it has not been thoroughly optimised. Further study is warranted, investigating approaches such as using a cyclic learning rate scheduler, smoothly altering sensor parameters throughout the curriculum and extending the final curriculum stage. In addition, a more rigorous evaluation should be conducted that freezes feature extractor weights in order to distinguish how much of the performance improvement stems from the curriculum's impact on attention as opposed to its impact on feature extraction.

### Predictive Attention

Evaluating the hardcoded attention spotlight in a more complex setting is the next logical step in the development of decomposed predictive attention. Such a setting can consist of an optimisation task guided by image classification loss rather than the summation loss that was described in Section 5.4.3; this would enable evaluating the efficacy of using HAS with real feature maps as opposed to simulated ones, as well as investigating the feasibility of applying HAS attention maps to the system's input space instead of the latent space.

When deployed in a more complex setting, the HAS parameterisation shown in Section 5.4.3 could result in a support Gaussian with an amplitude that is too high in the non-spotlight areas away from the specified coordinates. Future work should include investigating this possibility and searching for optimal HAS parameters. As it is a computationally cheap operation, its parameters can be computed at inference time in order to tailor the attention matrix to the input peripheral view; however, investigating this is left for future work.

The next research objective following a further investigation of HAS is developing a modelling mechanism capable of working alongside HAS to predict the foveal view at the next fixation location. Once such a mechanism is developed, the focus should shift to investigating different approaches to structuring the backward pass of predictive attention. Alternatively, the backward pass can be investigated if a novel active vision architecture and memory mechanism are devised, together with a suitable end-to-end predictive modelling mechanism.

The backward pass of predictive attention is an open-ended problem that can be structured in multiple different ways. What follows is a discussion of three proposed ways of solving this problem; note that this is not an exhaustive list.

The first method, dubbed the **imaginary approach**, is the most similar to the example shown in Figure 5.9 in Section 5.4.2 as it involves treating the predictive attention stream as an "imaginary" forward pass that runs independently from the real forward pass. As with the simplified example, the modelling mechanism predicts the foveal view using the next fixation coordinates and the peripheral view. The predicted foveal view is then passed to the classifier and this imaginary classification is compared against the real ground truth label of the image. The imaginary classification loss is then used to backpropagate gradients through the classifier, modelling mechanism and the fixation coordinates all the way into the attention mechanism. This approach is similar to that employed in the World Models paper published by Ha and Schmidhuber [107], where a model of the environment is used to train an agent's control policy.

The second method, dubbed the **energy-based approach**, does not utilize predictive attention gradients to learn an attention policy but instead uses them as the attention policy itself. Several researchers in deep learning use the term "energy" to refer to utilizing backpropagation at inference time in order to optimize a variable produced by or stored within the architecture [108], in contrast to using loss for optimizing the architecture's weights at training time. In this approach, the attention mechanism is replaced with a proposal network whose role is to suggest fixation coordinates that are only an initial estimate of the next viewing location. At the same time the classifier is upgraded to output a value representing its confidence in its prediction. As with the other approaches, the coordinates are passed to the modelling mechanism together with the peripheral view to predict a foveal view that is then passed to the classifier. The classifier's confidence output is used to compute energy for predictive attention; the energy is then used to backpropagate gradients into the proposed coordinates. The gradients update the coordinates so as to maximize the classifier's confidence in its prediction. Depending on the computational constraints imposed on the task, this approach can evaluate multiple initial fixation locations in parallel, finally selecting the fixation that results in the lowest energy cost. It is worth noting that calibrating the classifier's outputs enables it to produce a confidence metric, and that Gallos and Ferrie [77] have previously argued for the importance of calibrating outputs in active vision systems.

The final proposed method is dubbed the **desire-led approach**, or the loss-energy hybrid approach, and it more closely integrates the backward pass of predictive attention with the architecture's regular forward pass. In this approach the coordinates and the peripheral view are passed to the modelling mechanism to produce a prediction of the foveal view at the specified coordinates; however, in this case the predicted foveal view is not passed to the classifier. Instead, a copy of the true foveal view that the architecture has observed is passed to the classifier; the resulting classification output is compared against the image's ground truth label to produce classification energy[1]. This energy is then used to repeatedly backpropagate gradients into the copied foveal view, updating it until it results in the correct classification output when passed back into the classifier. The resulting updated foveal view is the *desired view*, and it is used as the ground truth label that the predicted foveal view is compared against to compute predictive attention loss. The loss is backpropagated through the modelling mechanism and into the attention module.

### 6.2.3 Complex Visual Tasks

Image classification is one of the simplest visual tasks that are being actively researched in deep learning, as it only demands the architecture to output an $N$-way decision, with $N$ being equal to the number of classes. This can be contrasted with object detection and segmentation, both of which are tasks that demand space-variant output that describes the entirety of the visual scene. In their widely cited publication that introduced the R-CNN architecture, Girshick et al. [109] claim that they "bridge the gap between image classification and object detection". This terminology is very valuable to guiding future active vision research, as no such approaches for bridging this gap have been formulated that are applicable to active vision. The R-CNN and the sliding window approaches that preceded it are antithetical to the active vision paradigm due to how strongly ingrained they are in the passive vision paradigm. Although currently this research direction might still be premature, future research should aim to develop mechanisms for active vision systems to complete visual tasks that require more complex outputs than classification.

[1]Energy is equivalent to loss that is being utilized during inference time, therefore what is meant by classification energy is classification loss that was repurposed to optimize a variable produced by the model rather than its weights.

# Glossary of Selected Terms

**appraisal model**

A feedforward passive vision architecture trained to perform image classification, used to appraise attention policies. When trained with an attention policy, the model's classification accuracy on the validation set is used to assign a value to the attention policy. Ideally, the appraisal model's architecture is as similar to the active vision model as possible to make the appraised values specific to the active vision model under study. , 65, 66, 68, 70–74, 89

**attention policy**

An algorithm that when provided with an image returns a singular fixation point. These can be combined to produce an attention policy sequence and produce multiple fixation points per image. , 63, 65–73, 77, 79, 81, 89, 90

**egocentric**

Within this dissertation an egocentric coordinate frame is one with the point of the active agent's first fixation as its origin. , 43, 50–52, 54

**energy**

A term commonly used in deep learning to refer to a value that is being optimized at inference time, as opposed to the loss function which is optimized only at training time. Energy optimisation is relevant to active vision attention as it can be utilized to obtain fixation coordinates without training a network that reliably produces good fixations with its forward pass. 124, 125

**exocentric**

Within this dissertation an exocentric coordinate frame is the input image's absolute coordinate frame, with the top-left of the image being the origin. , 43, 50–52, 58

**glimpse network**

A module in the Recurrent Attention Model architectures responsible for extracting all features from the visual and proprioceptive input obtained at the current time step. , 25, 26, 34, 47

**image vector**

A data structure containing colour intensity information that is associated with the receptive fields of a software retina, retinal image pyramid or a cortical mapping. The image vector is of shape $(C, N)$ where C stands for the number of colour channels and N stands for the number of receptive fields. 12–17, 19, 20

**SpaCat**

A feedforward active vision architecture with a memory mechanism that concatenates observations in image space. Stands in contrast to the Recurrent Attention Model architectures that maintain a latent memory tensor with an RNN. 94–97, 99–103

# Bibliography

[1] P. Ozimek and J. Siebert, "Integrating a Non-Uniformly Sampled Software Retina with a Deep CNN Model," in *BMVC 2017 Workshop on Deep Learning On Irregular Domains*, September 2017.

[2] E. L. Schwartz, "Spatial mapping in the primate sensory projection: Analytic structure and relevance to perception," *Biological Cybernetics*, vol. 25, no. 4, pp. 181–194, 1977. [Online]. Available: http://dx.doi.org/10.1007/BF01885636

[3] S. Balasuriya, "A computational model of space-variant vision based on a self-organized artifical retina tesselation," Ph.D. dissertation, Department of Computing Science, University of Glasgow, March 2006.

[4] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *International journal of computer vision*, vol. 1, no. 4, pp. 333–356, 1988.

[5] L. Mai, H. Le, Y. Niu, and F. Liu, "Rule of thirds detection from photograph," in *2011 IEEE International Symposium on Multimedia*. IEEE, 2011, pp. 91–96.

[6] P. Ammirato, P. Poirson, E. Park, J. Košecká, and A. C. Berg, "A dataset for developing and benchmarking active vision," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1378–1385.

[7] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz, "Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models," in *Advances in Neural Information Processing Systems*, 2019, pp. 9448–9458.

[8] A. Borji, "Objectnet dataset: Reanalysis and correction," *arXiv preprint arXiv:2004.02042*, 2020.

[9] D. H. Ballard, "Animate vision," *Artificial intelligence*, vol. 48, no. 1, pp. 57–86, 1991.

[10] V. Mnih, N. Heess, A. Graves *et al.*, "Recurrent models of visual attention," in *Advances in neural information processing systems*, 2014, pp. 2204–2212.

[11] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," *arXiv preprint arXiv:1412.7755*, 2014.

[12] Z. Li, Y. Yang, X. Liu, F. Zhou, S. Wen, and W. Xu, "Dynamic computational time for visual attention," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 1199–1209.

[13] D. Jayaraman and K. Grauman, "Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion," in *European Conference on Computer Vision*. Springer, 2016, pp. 489–505.

[14] ——, "End-to-end policy learning for active visual categorization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 7, pp. 1601–1614, 2018.

[15] P. Sermanet, A. Frome, and E. Real, "Attention for fine-grained categorization," *arXiv preprint arXiv:1412.7054*, 2014.

[16] R. Cheng, Z. Wang, and K. Fragkiadaki, "Geometry-aware recurrent neural networks for active visual recognition," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[17] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *arXiv preprint arXiv:2201.03545*, 2022.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[19] C. A. Curcio, K. R. Sloan, R. E. Kalina, and A. E. Hendrickson, "Human photoreceptor topography," *Journal of comparative neurology*, vol. 292, no. 4, pp. 497–523, 1990.

[20] D. H. Hubel, *Eye, brain, and vision.* Scientific American Library/Scientific American Books, 1995.

[21] U. S. Kim, O. A. Mahroo, J. D. Mollon, and P. Yu-Wai-Man, "Retinal ganglion cells—diversity of cell types and clinical relevance," *Frontiers in Neurology*, vol. 12, 2021.

[22] G. D. Field and E. Chichilnisky, "Information processing in the primate retina: circuitry and coding," *Annu. Rev. Neurosci.*, vol. 30, pp. 1–30, 2007.

[23] T. Gollisch and M. Meister, "Eye smarter than scientists believed: neural computations in circuits of the retina," *Neuron*, vol. 65, no. 2, pp. 150–164, 2010.

[24] R. Shapley and M. J. Hawken, "Color in the cortex: single-and double-opponent cells," *Vision research*, vol. 51, no. 7, pp. 701–717, 2011.

[25] T. N. Wiesel and D. H. Hubel, "Spatial and chromatic interactions in the lateral geniculate body of the rhesus monkey." *Journal of neurophysiology*, vol. 29, no. 6, pp. 1115–1156, 1966.

[26] S. Clippingdale and R. Wilson, "Self-similar neural networks based on a kohonen learning rule," *Neural Networks*, vol. 9, no. 5, pp. 747–763, 1996.

[27] C. Enroth-Cugell and J. G. Robson, "The contrast sensitivity of retinal ganglion cells of the cat," *The Journal of Physiology*, vol. 187, no. 3, pp. 517–552, 1966. [Online]. Available: http://dx.doi.org/10.1113/jphysiol.1966.sp008107

[28] T. Münch, S. Fried, and F. Werblin, "Starburst cells initiate directional selective responses in rabbit retina," *Investigative Ophthalmology & Visual Science*, vol. 43, no. 13, pp. 2981–2981, 2002.

[29] B. P. Ölveczky, S. A. Baccus, and M. Meister, "Segregation of object and background motion in the retina," *Nature*, vol. 423, no. 6938, pp. 401–408, 2003.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[31] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4161–4170.

[32] P. Ozimek, N. Hristozova, L. Balog, and J. P. Siebert, "A space-variant visual pathway model for data efficient deep learning," *Frontiers in Cellular Neuroscience*, vol. 13, p. 36, 2019.

[33] S. Gao, K. Yang, C. Li, and Y. Li, "A color constancy model with double-opponency mechanisms," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 929–936.

[34] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[35] R. W. Rodieck, "Quantitative analysis of cat retinal ganglion cell response to visual stimuli," *Vision research*, vol. 5, no. 12, pp. 583–601, 1965.

[36] J. Birch, “Efficiency of the ishihara test for identifying red-green colour deficiency,” *Ophthalmic and Physiological Optics*, vol. 17, no. 5, pp. 403–408, 1997.

[37] I. Rafegas and M. Vanrell, “Color representation in cnns: parallelisms with biological vision,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2697–2705.

[38] ——, “Color encoding in biologically-inspired convolutional neural networks,” *Vision research*, vol. 151, pp. 7–17, 2018.

[39] M. A. Cohen, T. L. Botch, and C. E. Robertson, “The limits of color awareness during active, real-world vision,” *Proceedings of the National Academy of Sciences*, vol. 117, no. 24, pp. 13 821–13 827, 2020.

[40] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, “Scaling egocentric vision: The epic-kitchens dataset,” in *European Conference on Computer Vision (ECCV)*, 2018.

[41] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, “A closer look at spatiotemporal convolutions for action recognition,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6450–6459.

[42] M. I. Posner, Y. Cohen, and R. D. Rafal, “Neural systems control of spatial orienting,” *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, vol. 298, no. 1089, pp. 187–198, 1982.

[43] P. S. Churchland, V. S. Ramachandran, and T. J. Sejnowski, “A critique of pure vision,” *Large-scale neuronal theories of the brain*, vol. 23, 1994.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[45] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” 2011.

[46] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, “Convolutional lstm network: A machine learning approach for precipitation nowcasting,” in *Advances in neural information processing systems*, 2015, pp. 802–810.

[47] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[48] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, 2017.

[49] A. Garm, M. Oskarsson, and D.-E. Nilsson, “Box jellyfish use terrestrial visual cues for navigation,” *Current Biology*, vol. 21, no. 9, pp. 798–803, 2011.

[50] D. Marr, “Vision: A computational investigation into the human representation and processing of visual information,” 1982.

[51] G. Kuhn, B. W. Tatler, and G. G. Cole, “You look where i look! effect of gaze cues on overt and covert attention in misdirection,” *Visual Cognition*, vol. 17, no. 6-7, pp. 925–944, 2009.

[52] N. M. Broomfield and G. Turpin, “Covert and overt attention in trait anxiety: A cognitive psychophysiological analysis,” *Biological Psychology*, vol. 68, no. 3, pp. 179–200, 2005.

[53] J. M. Findlay and I. D. Gilchrist, “Visual attention: The active vision perspective,” in *Vision and attention*. Springer, 2001, pp. 83–103.

[54] 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,” in *International conference on machine learning*, 2015, pp. 2048–2057.

[55] M. M. Chun and J. M. Wolfe, *Chapter nine visual attention*. John Wiley & Sons, 2008.

[56] L. Deng, “The mnist database of handwritten digit images for machine learning research [best of the web],” *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[57] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” 1999.

[58] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[59] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li, “Novel dataset for fine-grained image categorization: Stanford dogs,” in *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, vol. 2, no. 1. Citeseer, 2011.

[60] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, “3d object representations for fine-grained categorization,” in *Proceedings of the IEEE international conference on computer vision workshops*, 2013, pp. 554–561.

[61] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," 2011.

[62] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[63] M. Malmir, K. Sikka, D. Forster, J. R. Movellan, and G. Cottrell, "Deep q-learning for active recognition of germs: Baseline performance on a standardized dataset for active learning." in *BMVC*, 2015, pp. 161–1.

[64] M. Malmir, K. Sikka, D. Forster, I. Fasel, J. R. Movellan, and G. W. Cottrell, "Deep active object recognition by joint label and action prediction," *Computer Vision and Image Understanding*, vol. 156, pp. 128–137, 2017.

[65] E. Gärtner, A. Pirinen, and C. Sminchisescu, "Deep reinforcement learning for active human pose estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 10 835–10 844.

[66] R. Cheng, A. Agarwal, and K. Fragkiadaki, "Reinforcement learning of active vision for manipulating objects under occlusions," in *Conference on Robot Learning*. PMLR, 2018, pp. 422–431.

[67] S. Mathe, A. Pirinen, and C. Sminchisescu, "Reinforcement learning for visual object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2894–2902.

[68] L. Smith and M. Gasser, "The development of embodied cognition: Six lessons from babies," *Artificial life*, vol. 11, no. 1-2, pp. 13–29, 2005.

[69] J. Yang, Z. Ren, M. Xu, X. Chen, D. Crandall, D. Parikh, and D. Batra, "Embodied visual recognition," *arXiv preprint arXiv:1904.04404*, 2019.

[70] D. S. Chaplot, H. Jiang, S. Gupta, and A. Gupta, "Semantic curiosity for active visual learning," in *European Conference on Computer Vision*. Springer, 2020, pp. 309–326.

[71] K. Friston, J. Kilner, and L. Harrison, "A free energy principle for the brain," *Journal of physiology-Paris*, vol. 100, no. 1-3, pp. 70–87, 2006.

[72] K. Friston, "The free-energy principle: a unified brain theory?" *Nature reviews neuroscience*, vol. 11, no. 2, pp. 127–138, 2010.

[73] K. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, G. Pezzulo *et al.*, “Active inference and learning,” *Neuroscience & Biobehavioral Reviews*, vol. 68, pp. 862–879, 2016.

[74] T. Van de Maele, T. Verbelen, O. Catal, C. De Boom, and B. Dhoedt, “You only look as much as you have to: using the free energy principle for active vision,” in *IWAI2020, International Workshop on Active Inference*, vol. 1326. Springer, 2020, pp. 92–100.

[75] T. Van de Maele, T. Verbelen, O. Çatal, C. De Boom, and B. Dhoedt, “Active vision for robot manipulators using the free energy principle,” *Frontiers in neurorobotics*, vol. 15, p. 14, 2021.

[76] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, “Shapenet: An information-rich 3d model repository,” *arXiv preprint arXiv:1512.03012*, 2015.

[77] D. Gallos and F. Ferrie, “Active vision in the era of convolutional neural networks,” in *2019 16th Conference on Computer and Robot Vision (CRV)*. IEEE, 2019, pp. 81–88.

[78] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[79] B. Cheung, E. Weiss, and B. Olshausen, “Emergence of foveal image sampling from learning to attend in visual scenes,” *arXiv preprint arXiv:1611.09430*, 2016.

[80] H. Lukanov, P. König, and G. Pipa, “Biologically inspired deep learning model for efficient foveal-peripheral vision,” *Frontiers in Computational Neuroscience*, vol. 15, 2021.

[81] J. Martínez and L. A. Robles, “A new foveal cartesian geometry approach used for object tracking.” *SPPRA*, vol. 6, pp. 133–139, 2006.

[82] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[83] J. Schmidhuber and R. Huber, “Learning to generate artificial fovea trajectories for target detection,” *International Journal of Neural Systems*, vol. 2, no. 01n02, pp. 125–134, 1991.

[84] R. Monica and J. Aleotti, "A probabilistic next best view planner for depth cameras based on deep learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3529–3536, 2021.

[85] G. Elsayed, S. Kornblith, and Q. V. Le, "Saccader: Improving accuracy of hard attention models for vision," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[86] J. Fu, H. Zheng, and T. Mei, "Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4438–4446.

[87] C. Kyrkou, "C3 net: end-to-end deep learning for efficient real-time visual active camera control," *Journal of Real-Time Image Processing*, vol. 18, no. 4, pp. 1421–1433, 2021.

[88] H. MaBouDi, M. Roper, M. Guiraud, J. A. Marshall, and L. Chittka, "Automated video tracking and flight analysis show how bumblebees solve a pattern discrimination task using active vision," *bioRxiv*, 2021.

[89] S. Ravi, T. Siesenop, O. J. Bertrand, L. Li, C. Doussot, A. Fisher, W. H. Warren, and M. Egelhaaf, "Bumblebees display characteristics of active vision during robust obstacle avoidance flight," *Journal of Experimental Biology*, vol. 225, no. 4, p. jeb243021, 2022.

[90] M. Guiraud, "Pattern recognition and active vision in bees." Ph.D. dissertation, Queen Mary University of London, 2020.

[91] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order boltzmann machine," *Advances in neural information processing systems*, vol. 23, 2010.

[92] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[93] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[94] "Cats," Dec 2011. [Online]. Available: https://archives.evergreen.edu/webpages/curricular/2011-2012/m2o1112/web/cats.html

[95] H. T. Siegelmann, B. G. Horne, and C. L. Giles, "Computational capabilities of recurrent narx neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 27, no. 2, pp. 208–215, 1997.

[96] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, "Batch normalized recurrent neural networks," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 2657–2661.

[97] T. Cooijmans, N. Ballas, C. Laurent, Ç. Gülçehre, and A. Courville, "Recurrent batch normalization," *arXiv preprint arXiv:1603.09025*, 2016.

[98] M. A. Goodale and A. D. Milner, "Separate visual pathways for perception and action," *Trends in neurosciences*, vol. 15, no. 1, pp. 20–25, 1992.

[99] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[100] M. L. Courage and R. J. Adams, "Visual acuity assessment from birth to three years using the acuity card procedure: cross-sectional and longitudinal samples," *Optometry and vision science*, vol. 67, no. 9, pp. 713–718, 1990.

[101] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[102] A. C. Clayden, R. B. Fisher, and A. Nuthmann, "On the relative (un) importance of foveal vision during letter search in naturalistic scenes," *Vision Research*, vol. 177, pp. 41–55, 2020.

[103] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *arXiv preprint arXiv:1811.12231*, 2018.

[104] Q. Liao and T. Poggio, "Bridging the gaps between residual learning, recurrent neural networks and visual cortex," *arXiv preprint arXiv:1604.03640*, 2016.

[105] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[106] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. J. Rezende, "S3ta: A soft, spatial, sequential, top-down attention model," 2018.

[107] D. Ha and J. Schmidhuber, “Recurrent world models facilitate policy evolution,” *Advances in neural information processing systems*, vol. 31, 2018.

[108] Y. LeCun, “A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27,” 2022.

[109] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.