A VISUAL PIPELINE USING NETWORKS OF SPIKING NEURONS

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCIENCE AND ENGINEERING

2019

By
Garibaldi Pineda García
School of Computer Science

Contents

This thesis contains 34111 words.

List of Tables

List of Figures

Acronyms

AE Auto-Encoders.

AMPA Excitatory neurotransmitter whose effect is brief.

ANN Artificial neural network.

API Applications programming interface.

CNN Convolutional neural network.

CPU Central processing unit.

CRC Cyclic redundancy check.

DA Dopamine; modulatory neurotransmitter which is associated with reinforcement learning.

DVDT Voltage-change plasticity rule.

GABA$_B$ Inhibitory neurotransmitter whose effect is long-lasting.

GANs Generative adversarial networks.

GPU Graphics processing unit.

HH Hodgkin-Huxley neuron model.

LGN Lateral geniculate nucleus, part of the visual region in the thalamus.

LIF Leaky Integrate-and-Fire neuron model.

LTD Long-term depression, negative change done to synaptic efficacy..

LTP Long-term potentiation, positive change done to synaptic efficacy..

MDU Motion detection units.

MNIST MNIST database of handwritten digits, a subset of the original National Institute of Standards and Technology (NIST) dataset.

NMDA Excitatory neurotransmitter whose effect lasts a long time and requires the membrane potential to be at a certain level to be effective.

NVS Neuromorphic vision sensor.

RBM Restricted Boltzmann machine.

SAC Starburst amacrine cell.

SDR Sparse distributed representation.

SDRAM Synchronous dynamic random access memory.

SNN Spiking neural network.

SST SpiNNaker software tool-chain.

STDP Spike-timing-dependent plasticity.

SVD Singular value decomposition.

SWTA Soft winner-takes-all, a network circuit in which some neurons' activity inhibits some of their peers.

WTA Winner-takes-all, a network circuit in which a single neuron's activity inhibits all its peers.

Abstract

A VISUAL PIPELINE USING NETWORKS OF SPIKING NEURONS

Garibaldi Pineda García

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy, 2018

Computer vision has seen great advances recently due to artificial neural networks (ANNs) with multiple hidden layers; in particular, convolutional neural networks (CNNs) have shown great flexibility in the tasks they are able to perform. These networks are usually trained in a supervised manner using the back-propagation algorithm.

Spiking neural networks (SNNs) are the third generation of neuron simulation and have been proven to have better mathematical properties than previous generations. This approach to simulating biological neurons uses pulses to communicate the state of the units. The event-driven nature of SNNs imposes challenges for computer vision tasks. For example, neuromorphic vision sensors generate data when changes in their field of view are perceived; in contrast, standard cameras send full images. Furthermore, SNN training methodologies are not as mature as previous generations' procedures; critically, biologically-plausible supervised learning algorithms are scarce.

In this thesis we explore event-based computer vision using spiking neurons as computation units; the main goal is to build a fully-spiking visual pipeline. Firstly, we develop methods to transform standard images into spike representations. We then process the generated spike trains to extract salient features using biological principles. Finally, supervised learning algorithms for SNNs are developed and applied in a computer vision context.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

The experience I had towards obtaining a doctoral degree has been mentally stimulating and challenging. I wholeheartedly express my gratitude to my supervisor Professor Steve Furber for his patience and the interesting discussions as, without them, this thesis would not have been attainable.

I want to thank the people of México whose contributions allowed the National Council for Science and Technology (CONACyT) to sponsor my PhD studies.

I would like to show my deepest appreciation to my colleagues John V. Woods, Simon Davidson, Michael Hopkins, Luis Plana and Jim Garside for the diverse topics we had the chance to discuss. Their knowledge has nurtured my understanding of a broad range of topics. My eternal gratitude goes to James Knight, Alan Stokes, Andrew Rowley and Oliver Rhodes for their help in disentangling my view of the SpiNNaker software stack. Many thanks to my lab partners Qian Liu, Petruţ Bogdan, Mantas Mikaitis, Robert James, Patrick Camilleri and James Knight for the talks and collaborations which made my research possible.

The path towards the PhD would have not been bearable without the presence of my friends. I would like to thank Qian Liu, Mireya Paredes López, Valentina Ramírez Paredes, Thanos Stratikopoulos, Crefeda Rodrigues, Petruţ Bogdan, Mantas Mikaitis, Robert James, Patrick Camilleri, James Knight, Nuno Miguel Nobre, Gengting Liu and Raúl García for all the joy you brought to my life.

I want to thank my family for their love, support and encouragement towards this goal. My parents and siblings have shown great understanding and acceptance of my quirks; for this I am forever grateful. My nephews and nieces have given me great moments of happiness and joy which lift my mood immediately. I specially want to express my love and gratitude to my wife whose company, love and care have kept me going through this journey. This degree is partly hers.

Chapter 1

Introduction

Vision is the primary sense for many modern human activities, from driving to reading. In fact, humans have even modified the environment for rapid visual information acquisition by, for example, adding white strips to designate safe street-crossing areas or land-slide sign posts near some mountains. Given this existing infrastructure allowing rich interaction with the world it would be convenient to have a machine which could make use of it for tasks such as navigation. Research on emulating the human visual system has been ongoing since around the 1960s [Szeliski, 2010]. To this day, computer vision systems have come close to human performance in specific tasks, but general-purpose artificial vision remains an open question. Some visual tasks from static input are object detection, recognition, localization and identification; whether the object is present, what and where in the image it is, and what instance of the category it is (e.g. a human in the animal category). These usually require modelling a function which maps from raw pixel data to required information (e.g. categories).

Recent advances in computer vision base their success on Artificial Neural Networks (ANNs), particularly Convolutional Neural Networks (CNNs) [Goodfellow et al., 2016]. This is mainly due to the continuing increase in computational resources and training data; for example digit-recognition networks can be trained in hours while crunching through 60,000 labelled samples. These models are typically trained and executed on Graphic Processing Units (GPU); while the parallel nature of GPUs allows them to accelerate computations, the hardware is not fully compatible with neural networks, thus energy efficiency is low.

A hardware platform which supports biological neural network architectures should allow us to execute ANNs –and vision tasks– in an energy-efficient way. SpiNNaker is a system specifically constructed to simulate Spiking Neural Networks (SNNs); it

uses multiple low-power ARM cores arranged in chips with shared memory, every chip can connect to 6 neighbours and communication is asynchronous [Furber et al., 2013]. Using standard Central Processing Units (CPUs) as computing units provides an excellent platform for research as it is possible to program different neuron, synapse and plasticity models. Since the SpiNNaker machine is also a low-power device, it is suitable for mobile applications such as robots.

Computer vision relies on static images (or sequences of them) which is very different from biological vision where sensors emit fewer signals, mainly of changes occurring in the environment. Researchers have shown that collecting statistics on event-based sensors, which are similar to biological sensors, can lead to full visual reconstruction [Kim et al., 2016]. Something similar may be happening in the human cortex where plasticity could be forming a model of the world which is perceived as a static image. Given that mammalian vision is still superior to its machine counterpart, we can use it as an inspiration to develop a pipeline which can solve many vision tasks. The components of the visual pipeline in mammals are: eyes (sensors), thalamus (flow control/attention) and cortex (modelling) [Hubel et al., 1995]. Each of these components has been specialized through evolution; moreover, each can be thought of as an independent brain, thus simulating them is a complex task. Cells in the retina, the neural structure in the eye, communicate through local voltage gradients and, at the output, ganglion cells emit impulses known as spikes. In the thalamus and cortex most communication is done through spikes and computation is thought to be local and a function of these signals.

Visual processing using event-based sensors is a relatively new field and performing the required computation with spiking neurons imposes additional challenges. The work presented here aims to explore the requirements and possibilities of a biologically-inspired, fully-spiking visual processing pipeline. In mammals, nature has run an experiment to make robust and energy-efficient vision organs. We will extract principles from these and use them to generate visual pipeline components.

In this thesis we explore event-based computer vision and make use of biological principles to develop a vision pipeline making use only of SNNs. Using the SpiNNaker machine for this purpose imposes challenges; this work will address these demands. We will build a biologically-plausible, SNN-based computer vision pipeline (Figure 1.1) and evaluate it at each stage. For example, when converting images or video to spike representations we compare the activities of pixels. Another example is that, at the second stage of the pipeline, we measure the accuracy of the responses

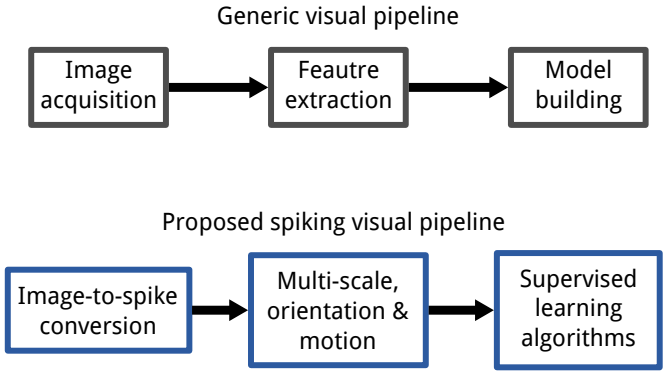


Figure 1.1: Comparison of visual pipelines.

from direction selective circuits.

Contributions

The main contributions of this Thesis are:

- **Image to spike representation conversion.** A large collection of visual data has been acquired by the traditional computer vision community. A way to make use of these resources for SNN research is to convert them into a spiking representation using software techniques; this has the additional benefit of establishing a way to compare standard computer vision techniques with their spiking networks equivalents. We present various methods to convert static images and video into spiking representations.

- **Visual processing building blocks and efficient deployment**. State of the art visual systems extract features from raw images. We develop spiking versions of edge and oriented bar detection, multi-scale representation, and motion sensing algorithms. We present a methodology for increasing the speed at which neural networks with distance-dependent connectivity are sent to the SpiNNaker machine. This methodology can be used to quickly deploy the networks used for feature extraction.

- **Supervised learning algorithms for spiking neural networks.** To collect the statistics of incoming stimuli, neural networks modify their synaptic weights. This is done based on plasticity rules; we demonstrate voltage-based and modulated spike-timing-based rules.

Publications

Most of the work which led to the content of this Thesis has been published in the following articles:

- **Benchmarking Spike-Based Visual Recognition: A Dataset and Evaluation** [Liu et al., 2016]. *Liu, Q., Pineda García, G., Stromatias, E., Serrano-Gotarredona, T., and Furber, S. B. (2016). Frontiers in Neuroscience.* In this paper we propose multiple algorithms to convert standard image datasets into spiking representations. The techniques presented are the following: rate-based Poisson spike generation, rank order encoding, and recorded output from a silicon retina with both flashing and oscillating input stimuli. An evaluation methodology is also shown to establish fair comparison of model and hardware performance. Finally, different network architectures are evaluated using rate-based input and the proposed methodology. The author contributed to this paper by implementing the Rank-order encoding and writing about it; additionally, we performed general repository management.

- **PyDVS: An extensible, real-time Dynamic Vision Sensor emulator using off-the-shelf hardware** [Pineda García et al., 2016]. *Pineda García, G., Camilleri, P., Liu, Q., and Furber, S. (2016). 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pages 1–7.* Neuromorphic Vision Sensors (NVSs) have excellent technical capabilities, though they remain scarce and relatively expensive. We propose a system inspired by the behaviour of an NVS but using a conventional digital camera as a sensor and a PC to encode the images. While our primary goal is to provide spiking neural networks with a live, real-time input, we have also been successful in transcoding well established image and video databases into spike train representations. The contribution of the author in this paper was the development of an NVS emulator and writing about it.

- **Neuromodulated Synaptic Plasticity on the SpiNNaker Neuromorphic System** [Mikaitis et al., 2018]. *Mikaitis, M., Pineda García, G., Knight, J. C., and Furber, S. B. (2018). Frontiers in Neuroscience.* SpiNNaker is an excellent tool for investigating multiple aspects of SNNs, among them synaptic plasticity which is thought to be a key mechanism behind learning. Standard Spike-Timing-Dependent Plasticity (STDP) rules have already been implemented on SpiNNaker. Unfortunately, STDP is an unsupervised learning mechanism, so

reinforcement signals require modification of the rule. To do so, we add a third factor to the plasticity algorithm which modulates weight changes. This paper shows the implementation of dopaminergic interaction with STDP on the SpiNNaker system and demonstrates it with a Pavlovian conditioning experiment. In this paper, the author's contribution was the conditioning experiment as well as optimizations to the SpiNNaker implementation which allowed the experiment to execute in real time using large populations (greater than a thousand neurons).

- **Spiking neural networks for computer vision** [Hopkins et al., 2018]. *Hopkins, M., Pineda García, G., Bogdan, P. A., and Furber, S. B. (2018). Interface Focus of the Royal Society.* In this paper we demonstrate the use of SNNs for computer vision using event-based processing on the SpiNNaker machine. Among other networks we show the conversion of NVS input to a multi-scale representation for spike count reduction. Making use of the flexibility offered by SpiNNaker we develop a spike-based motion detection network. Additionally, we explore structural synaptic plasticity as a mechanism to learn input statistics. The author contributed to this work with the multi-scale representation network and the motion sensing neural network.

Thesis structure

Chapter 2 provides an introduction to artificial neural networks and, particularly, to spiking neuron and synapse models, some of which will be used in later chapters. Additionally, we provide an overview of neuromorphic hardware. Since most of the experiments in this thesis were run on the SpiNNaker machine (except for the image acquisition section), we give special attention to a review of the SpiNNaker machine.

Chapter 3 provides an overview of vision in mammals, from which we obtain inspiration to develop the networks presented in this document. The image processing section of the pipeline follows connectivity found in the mammalian retina; while the model-building stage has closer resemblance to computational models of the cerebral cortex.

Most models of visual processing make use of distance-dependent connectivity; crucially, convolutional networks repeat weights and load times can be greatly increased. In Chapter 4 we present a solution to efficiently upload networks with this characteristic to the SpiNNaker machine.

Chapter 5 presents visual processing networks inspired by retinal and thalamic circuitry. The connectivity required by these networks can be efficiently loaded to the SpiNNaker machine via the optimizations presented in Chapter 4. These networks perform standard computer vision operations such as image filtering, multi-scale representation and motion detection.

In addition to these vision operations we would like to have a system which learns the statistics of its inputs. To learn, plasticity algorithms are required to adjust the weights in an SNN; supervised versions of these procedures are studied in Chapter 6. Finally, a summary of the research and suggestions for future work are presented in Chapter 7.

Summary

A general description of the research presented in this thesis was given in this Chapter, along with concrete contributions, publications and short description of its structure. The following Chapter provides a review of spiking neural network basic components: neurons and synapses. We will also give an overview of the SpiNNaker machine, a custom hardware platform to simulate SNNs using standard digital processors.

Chapter 2

Spiking neural networks

In this Chapter we present background knowledge on artificial neural networks which are the computing elements for most simulations presented in this Thesis. We also provide a review of the SpiNNaker massively-parallel machine designed to simulate spiking neural networks (SNNs) efficiently. Research on artificial neural networks has focused on an abstraction of biological neurons where units transmit continuous values which are thought to represent the rate of activity of biological neurons. Spiking neural networks behave closer to their biological counterparts than conventional ANNs and are considered the state of the art. Simulating SNNs on electronic hardware can be efficient and, sometimes, execute tasks at faster than real-time speed. In particular, experimenting with network architectures, neural behaviour and learning algorithms can be easier on programmable digital hardware than a static –or configurable– hardware simulators.

Neuron models

Mathematical models for a neuron's membrane potential behaviour started appearing in the first half of the 20th century [Brunel and Van Rossum, 2007]. They range from extremely detailed ones that consist of several differential equations to simple ones with just one or two; and they all model the electrical properties of nerve cells. A particular group of models describes the neuron as an equipotential sphere, that is, all its surface has the same electrical potential (Figure 2.1) [Dayan and Abbott, 2001].

Since the neuron is modelled as a sphere, the membrane *capacitance* C_m and *resistance* R_m are specified in relation to its area A.

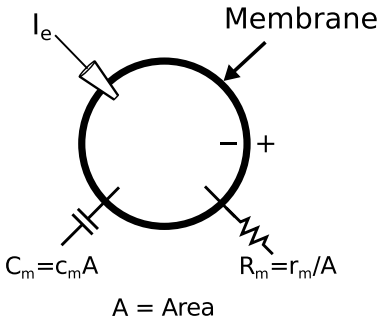


Figure 2.1: Diagram of the isopotential neuron. Adapted from Dayan and Abbott [2001].

$$R_m = r_m/A \tag{2.1}$$

$$C_m = c_m A \tag{2.2}$$

where r_m and c_m are the resistance and capacitance per unit area, respectively. Their values are $r_m \approx 1M\Omega mm^2$ and $c_m \approx 10nF/mm^2$. The basic relations of the electrical properties of the membrane shown in Figure 2.1 are as follows:

$$V = I_e R_m \,, \tag{2.3}$$

$$Q = C_m V \,, \tag{2.4}$$

$$C_m \frac{dV}{dt} = \frac{dQ}{dt} \,. \tag{2.5}$$

When the membrane's potential (V) changes, it does so according to Eq. 2.3. Variable Q is the membrane's electrical charge which is proportional to the product of its voltage and capacitance, as seen in Eq. 2.4. Currents originated by ion channels are approximated to a linear behaviour, and can be modelled using Ohm's law

$$i_x = g_x(V - E_x) \tag{2.6}$$

where E_x is the *reverse potential* due to ion exchange in channel x and g_x is the per unit area conductance of the channel. The total membrane current due to channels (per unit

area; i_m) will be

$$i_m = \sum_x i_x = \sum_x g_x(V - E_x) \tag{2.7}$$

To obtain the total membrane current (I_m) due to ion channels, Eq. 2.7 has to be multiplied by the total area A.

$$I_m = i_m A \tag{2.8}$$

The right hand side of Eq. 2.5 is the total current in the membrane. Since we are adding an external current I_e of opposite direction to I_m, the total current in the membrane is

$$I_T = I_e - I_m \tag{2.9}$$

Combining equations 2.5 and 2.9 with the fact that $i = dQ/dt$, gives the basic equation used by most single-compartment models [Dayan and Abbott, 2001].

$$C_m \frac{dV}{dt} = I_e - I_m \tag{2.10}$$

$$c_m \frac{dV}{dt} = \frac{I_e}{A} - i_m \tag{2.11}$$

Hodgkin-Huxley model

In 1952, Hodgkin and Huxley published a paper that reflected their ground-breaking experimental work on the axon of the squid [Hodgkin and Huxley, 1952]. They found that the currents in the membrane are mainly due to changes in the concentration of three ions: potassium (K^+), sodium (Na^+) and chlorine (Cl^-). The first two are related to voltage dependent conductance and the last to a leakage current. Under these considerations [Izhikevich, 2007a], Eq. 2.10 becomes:

$$C_m \frac{dV}{dt} = I_e - g_K n^4 (V - E_K) - g_{Na} m^3 h (V - E_{Na}) - g_L (V - E_L) \tag{2.12}$$

the functions n, m and h have the following behaviour

$$\frac{dn}{dt} = \alpha_n(V)(1-n) - \beta_n(V)n \,, \tag{2.13}$$

$$\frac{dm}{dt} = \alpha_m(V)(1-m) - \beta_m(V)m \,, \tag{2.14}$$

$$\frac{dh}{dt} = \alpha_h(V)(1-h) - \beta_h(V)h \,; \tag{2.15}$$

these represent the activation and inactivation dynamics of the potassium, sodium and chlorine channels, respectively.

The behaviour of α_x and β_x in Equations 2.13, 2.14 and 2.15 is given by:

$$\begin{aligned} \alpha_n(V) &= 0.01\frac{10-V}{\exp^{(10-V)/10}-1}\,, & \beta_n(V) &= 0.125\exp^{-V/80}\,, \\ \alpha_m(V) &= 0.1\frac{25-V}{\exp^{(25-V)/10}-1}\,, & \beta_m(V) &= 4\exp^{-V/18}\,, \\ \alpha_h(V) &= 0.07\exp^{-V/20}\,, & \beta_h(V) &= \frac{1}{\exp^{(30-V)/10}+1}\,, \end{aligned} \tag{2.16}$$

these functions represent how channels change from an open and closed state and are characterized to have a resting membrane potential (V) at 0mV. Per channel potentials at equilibrium state are:

$$E_K = -12\text{mV}, \quad E_{Na} = 120\text{mV} \quad \text{and} \quad E_L = 10.6\text{mV}\,. \tag{2.17}$$

Finally, the maximum conductance values are commonly set to

$$g_K = 36\text{mS/cm}^2, \quad g_{Na} = 120\text{mS/cm}^2 \quad \text{and} \quad g_L = 0.4\text{mS/cm}^2\,. \tag{2.18}$$

The Hodgkin-Huxley (HH) model is one of the most detailed so far; it has served as the inspiration and foundation of many studies. This model is computationally expensive, so digital hardware to simulate it in real-time would require high energy consumption.

Leaky Integrate-and-fire model

The Leaky Integrate-and-Fire (LIF) model is one of the oldest neuron models, but is still used due to its simplicity and low computational cost. In the passive *integrate-and-fire* all the membrane conductances are modelled by a single term G_L.

$$C_m\frac{dV}{dt} = I_e - G_L\,(V - E_L) \tag{2.19}$$

$$C_m\frac{dV}{dt} = I_e - G_L V + G_L E_L \tag{2.20}$$

E_L is the reversal potential of the neuron and the rightmost term ($G_L E_L$) is also known as the *leak current* which pushes the membrane potential to the reversal value. If

Eq. 2.20 is multiplied by the membrane resistance R_m, we obtain

$$\tau_m \frac{dV}{dt} = R_m I_e - V + E_L \tag{2.21}$$

where τ_m is called the membrane time constant and can be interpreted as the rate at which the neuron moves towards its resting state. Integrating Eq. 2.21 results in an expression for the voltage behaviour under non-spiking conditions.

$$V(t) = E_L + R_m I_e + (V(0) - E_L - R_m I_e)\, e^{-t/\tau_m} \tag{2.22}$$

Spiking behaviour is added artificially once $V(t)$ reaches a certain threshold, afterwards it is reset back to $V(0)$. This is also set to a special value (V_{reset}), usually lower than the resting potential, in most SNN simulations.

Since it is simpler to characterize the LIF model we will use this to develop the networks which do visual processing found in Chapter 5.

Izhikevich model

The dynamics of the Hodgkin-Huxley model were studied using bifurcation diagrams and approximated by FitzHugh [1961]. Using similar ideas and techniques, Izhikevich developed what he named the *simple model* of spiking neurons [Izhikevich, 2003]. This model emulates the dynamics of the membrane voltage in the sub-threshold area and it consists of a pair of equations:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u - I \tag{2.23}$$

$$\frac{du}{dt} = a(bv - u) \tag{2.24}$$

where v represents the membrane voltage and u a negative feedback to v. The rising part of the spiking behaviour is produced by the equations, though an artificial voltage reset is needed afterwards. When variable v reaches 30mV or more, variables v and u are set as follows:

$$v = c\,, \tag{2.25}$$

$$u = u + d\,. \tag{2.26}$$

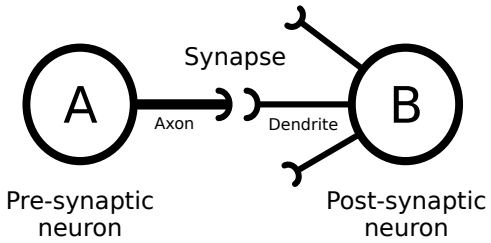


Figure 2.2: Neuron A can send a message to neuron B through a synapse.

where parameters a, b, c and d are dimensionless. Changes in the values of the parameters result in different neuron responses [Izhikevich, 2007a]. Izhikevich's model is a better approximation to the HH model, when compared to the LIF model described above, and the computational cost is reasonably low [Izhikevich, 2004]. It is the dynamics of this neuron model which incited us to use this model to test the voltage change-based rule presented in Chapter 6.

Synapse models

As we have described, information processing elements in the nervous system are known as neurons, which are cells that posses branch-like structures (axons and dendrites) enabling fast and long-distance communication, when compared to other cell types in the body. To communicate to other neurons, signals need to traverse the axon of a source neuron (A in Figure 2.2), then cross a channel known as a *synapse* (middle of Figure 2.2), and, finally travel through the dendritic tree in the target neuron (B in Figure 2.2) to reach its body (soma).

We take special interest in the current produced by the reception of different neurotransmitters in the post-synaptic neuron [Gerstner et al., 2014]. This way, the total input current to a (*post*) neuron is given by the sum of the currents provided by these channels,

$$I_{in} = \sum_{i}^{N_{pre}} I_{i,post} \ . \tag{2.27}$$

There are a couple of ways to model synapse behaviour, the simplest is to equate

synaptic efficacy ($w_{pre,post}$) and the current a spike will provide to the destination neuron:

$$I_{pre,post}(t) = w_{pre,post} g_{pre,post}(t) \tag{2.28}$$

where the function $g_{pre,post}(t)$ can generate shapes for the post-synaptic current (e.g. delta, exponential, multi-exponential).

The second method to simulate synapse behaviour is to model the conductance of ion channels

$$I_{pre,post}(t) = w_{pre,post} g_{pre,post}(t)(u(t) - E_{syn}) \, . \tag{2.29}$$

Similarly to the previous case $g_{pre,post}(t)$ serves as a current-shaping function, $u(t)$ is the membrane potential and E_{syn} is the synaptic reversal potential. The interaction between the two last quantities is the main difference, it can be seen as a saturation mechanism for input currents.

Encoding information with spikes

Neurons in the mammalian brain use voltage pulses (spikes) to communicate with each others. It's thought that these spikes encode information, though the encoding is still unknown. There are many hypotheses of how neurons encode information in spike trains, we will explain the two main ones: rate and temporal encodings [Dayan and Abbott, 2001; Gollisch, 2009].

Rate code

Spike-rate encoding, in its simplest case, represents information with the number of action potentials that a neuron generate in a time interval. It is one of the earliest attempts to explain neural encoding and gives a nice transition from traditional artificial neural networks to spiking ones. Furthermore, there is evidence that neurons in direct contact with sensory or motor organs encode information in this way (i.e. the stronger a muscle is flexed, the higher the rate of spikes generated by neurons near muscular tissue) [Brunel and Van Rossum, 2007].

Given a time period T and a spike resolution of t_s, we can encode, at most, $n_v = \lfloor T/t_s \rfloor$ values. In this way we discard any temporal information embedded, no matter

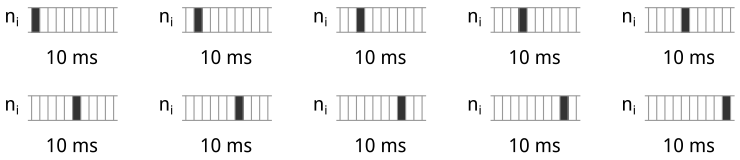


Figure 2.3: Spike rate example, all 10 examples encode the same value for they all have a 1 spike per 10ms window.

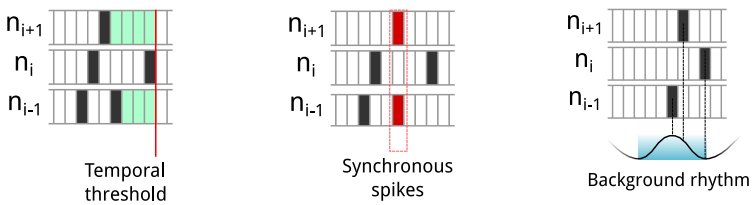
when the spike was received the encoded value will still be the same. Figure 2.3 shows the case of a 100Hz signal observed in a 10ms window.

Emitting a spike can be considered costly energy-wise, so using rate coding everywhere in the brain would require high energy, but the brain is rated at approximately 20 watts. Moreover, neurons would have to wait until sufficient events are received to distinguish between rates; this would "slow-down" reaction times and be a disadvantage from a survival point of view.

Temporal codes

In temporal encoding the precise time a spike is emitted reflects the input value; this translates into a larger representation capacity [VanRullen et al., 2005]. Although the precise time is a popular hypothesis, it is unlikely that temporal precision is part of neural computation, at least on a millisecond scale. The term temporal code encompasses different encoding techniques, some of which are briefly introduced bellow.

Time-to-first-spike. Information is encoded in the time it takes a neuron to spike after a certain temporal barrier is set (Figure 2.4a). The value is usually greater if the time between the barrier and spike is shorter.



(a) Time-to-first-spike (b) Synchrony (c) Phase

Figure 2.4: Different temporal codes

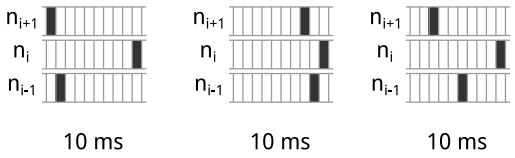


Figure 2.5: Rank-order encoding example. Since all the examples have neurons firing in the same order, they all encode the same value.

Synchrony. Whenever neurons fire at the same time, they should be classified together. A value would be represented by different combinations of neurons firing at the same time (Figure 2.4b).

Phase. The values are encoded in the phase of the spikes with respect to a background oscillating signal or rhythm (Figure 2.4c). This is similar to the time-to-first-spike encoding, but values can also be modulated by the background signal.

Rank-order. The temporal order of the spikes, and not their emission time, is what determines the value. Notice that it is also necessary to have a temporal reference as with the time-to-first-spike approach.

An issue that has been noted on rank-order encoded information is the fact that spike trains that are notably different (Figure 2.5) are interpreted as the same value. This might be corrected by changing the metrics used to determine the uniqueness of a spike train set [Cessac et al., 2010]. On the other hand, this same issue could be seen as a robust way of encoding information.

Neural simulation and neuromorphic hardware

While computers have reached great processing power, they require high energy consumption to achieve it. Furthermore, achieving human-level performance in certain tasks (e.g. pattern recognition, inference, puzzles) has pushed the use of Graphic Processing Units (GPUs) as parallel, general-purpose processors to train neural networks. Although GPUs are now considered highly parallel computing platforms and could be seen as a natural fit for neural networks, their data communication strategies are of a different nature. These processing units were built to deal with streaming data in parallel, particularly single pixels or vertices loaded from contiguous regions in memory. Neural networks, in contrast, require non-contiguous data; thus the efficiency is

usually lower than expected. Moreover, GPUs consume large amounts of energy with high performance versions utilizing hundreds of watts [Wikipedia, 2017a,b].

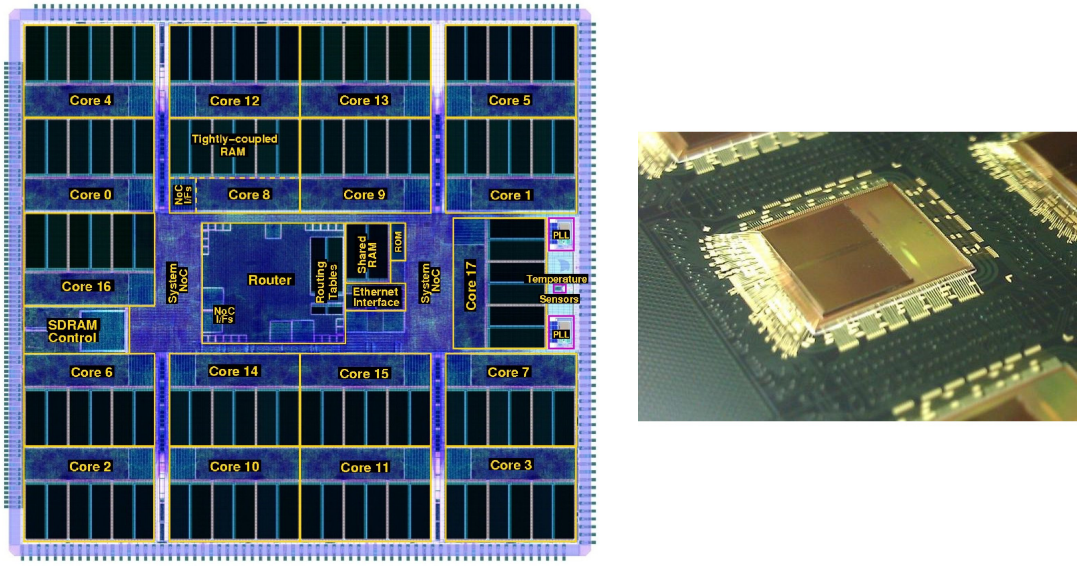
In comparison, human brains are estimated to consume about 20 watts while performing learning and inference simultaneously. Emulating biological computation circuitry could result in energy-efficient and near human-level performing systems. The term *neuromorphic* (i.e. that resembles neural form) is attributed to Mead and Ismail, probably coined as they worked on the implementation of silicon retinas [Mead and Ismail, 2012; Mead and Mahowald, 1988]. The main requirements for hardware to be considered neuromorphic are: low power consumption, real-time functionality, scalability and fault tolerance. Further work on neuromorphic sensors has led to silicon retinas, cochleas and visual motion sensors, to name a few [Liu and Delbruck, 2010]. While sensory input is of utmost importance for every system, a platform to make use of these sensors for AI tasks is still an open research problem.

Neuromorphic hardware platforms which combine the knowledge acquired from neuroscience and artificial neural networks research have recently been developed; they may be classified based on the type of hardware/software combination used. Hardware neurons are analogue circuits that behave like a mathematical model of a neuron; software neurons simulate the model using a general digital processor. Similar distinctions can be made for synapses, axonal and dendritic trees [Liu et al., 2016; Misra and Saha, 2010].

The SpiNNaker project.

The *SpiNNaker* project aims is to build a massively-parallel, bio-inspired, neuromorphic computing platform; the largest machine will contain $\sim$1 million CPU cores and will be able to simulate $\sim$1 billion neurons in real time. As for any multi-core system, communication between cores is fundamental; the SpiNNaker chip has been designed to implement spiking neural networks efficiently, thus networking is specialized (but not limited) to transmit spikes. There are currently two board models the Spinn-3, which hosts 4 SpiNNaker chips, and the Spinn-5 that houses 48 nodes. These boards are also known as 102 and 103 machines, respectively. Larger systems are built using Spinn-5 boards as the basic construction block:

104 machine. A desktop frame that has 12 Spinn-5 boards, which translates into $\sim$10,000 ARM cores.



(a) (b)

Figure 2.6: SpiNNaker base component. a) SpiNNaker chip die. b) Stitch-bonded SDRAM on top of the SpiNNaker chip.

105 machine. A server-like machine built using five card frames, each with 24 Spinn-5 boards in them. They will all be allocated in a rack cabinet, this machine has ∼100,000 cores in it.

106 machine. The largest machine is still work in progress and will be composed of 10 racks, each with similar capacities to the 105 machine (∼1,000,000 cores).

SpiNNaker chip overview

SpiNNaker chips contain 18 ARM968 cores, each with tightly-coupled program and data memory. Each core has a Communication Network-on-Chip (C-NoC) interface that, with the help of an in-die network router, allows inter- and intra-chip communications (Figure 2.6a). On top of the chip lies 128 MBytes of Synchronous Dynamic Random Access Memory (SDRAM) whose die is stitch-bonded to the chip for faster access (Figure 2.6b).

One of the cores acts as a system controller and, typically, 16 cores will run user applications. Any processor in a chip will have access to four memory spaces. Instruction and Data memories are local-only spaces (i.e. per-core memory) and composed of a 32- and a 64-KByte tightly-coupled blocks of memory, respectively. There are also a

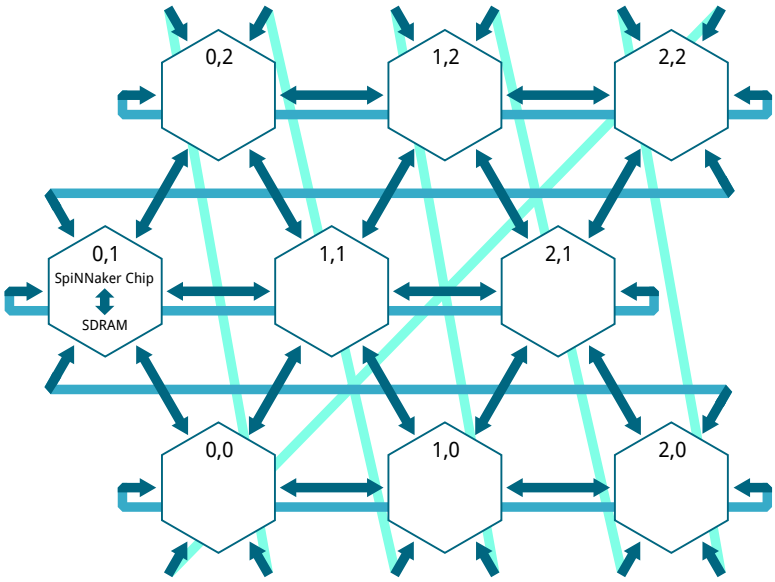


Figure 2.7: Nine SpiNNaker nodes connected as a toroid.

32 KByte block of on-chip Static RAM (SRAM) and the 128 MByte off-die SDRAM, both of which are node-local (i.e. seen by all cores in the chip). The cores are able to access the SDRAM and SRAM using the System Network-on-Chip (S-NoC) and no inter-chip memory coherence mechanisms are present [Furber et al., 2013]. SDRAM is commonly used for large data structures such as the synaptic weight matrices, while the SRAM is used for core-to-core message passing.

Communications

One of the notable aspects of the SpiNNaker system is the flexibility of its networking infrastructure. Communication in the global scale is asynchronous and locally synchronous. There are six direct links from any source node to its neighbours; this provides sufficient connections to form an efficient network that may be configured to form a toroid (Figure 2.7) on a 2D circuit board.

Packets are the only way a SpiNNaker chip can communicate with another; a core sends a packet to the local router where it is forwarded to one or more targets. If the target is in the same node, the router will deliver it directly; otherwise, the router sends the packet to an adjacent node whose router will decide the appropriate move. Packets are either 40- or 72-bits long, consisting of a control byte and one or two 32-bit words.

There are four packet types, each of which is routed differently by the router. *Nearest neighbour* (NN) and *point-to-point* (P2P) packets may be generated by any core, but will only reach the monitor core on the target node. NN packets are mainly used for boot and recovery and P2P for code distribution and system control. *Fixed route* (FR) packets may be emitted by any core and sent to any other core, though the route may not be changed once an application is running. This packet type is generally used for debugging purposes. The *multicast* (MC) packet also permits core-to-core transmission, so user applications use this type to transmit data. If the packet has multiple targets, a router along the way will duplicate it and fork the route. This is a highly desired feature for neural applications, since one neuron's axon may connect to multiple neurons' dendrites [Patterson et al., 2012]. MC packets are formed using address-event representation (AER), so they posses a time stamp and a full source address (node, core, and neuron identifications).

Communication to the outside world is done through 100 Mbit/s Ethernet, General Purpose Input/Output (GPIO) lines and through Field-Programmable Gate Arrays (FPGAs, on the 48-node boards).

Programming modes

Since neurons are thought to react as spikes arrive at their dendrites, SpiNNaker chips support event-based computation triggered by specialized interrupt hardware. There are, primarily, two choices while trying to program SpiNNaker.

The first is through the *Spin1* applications programming interface (API) following an event-driven model. In this way applications need specify only what functions to execute when an event occurs. Events include multicast packet reception, direct memory access (DMA) completion, timer ticks, SpiNNaker datagram protocol packet arrival, and custom user events. Using this type of programming allows developers to create any application and not only neural network simulations [Project, 2018].

The second is to use the neural network interface PyNN [Davison et al., 2009], which provides a common front-end to different simulators (e.g. NEST, Brian, sPyNNaker). Since the descriptions are stated using the Python programming language and because it is a high level description, the learning curve for this approach is moderate [Project, 2018]. SpiNNaker's PyNN interface is built on top of the Spin1 API, and it launches applications on the system to perform the computations that have been requested in the PyNN script. A common PyNN application would set-up the simulator parameters, describe neural populations, establish the appropriate projections and start

the simulation. The sPyNNaker team has added modules for real-time interaction with PyNN scripts and facilities for adding new neural and synaptic models.

Summary

This chapter reviewed spiking neuron models of varying levels of accuracy; a real-time simulation will likely benefit from less complex models since they require fewer computational resources. Spiking neural networks are more flexible than traditional artificial neurons, in particular the temporal dynamics can easily be configured and used for time-varying inputs. Furthermore, if information is represented using temporal encoding, these dynamics should help produce a more efficient system.

An introduction was also given to neuromorphic hardware which takes inspiration from biology to reduce power consumption; the SpiNNaker case was further described. The SpiNNaker machine can be seen as a software-based neural simulator, it is flexible both in terms of model simulation and network architecture. The main trade-off is that although it is suitable for experimentation, it may not be as power-friendly as hardware-based solutions.

The next chapter reviews the general structure of the visual pathway and describes different models for each stage.

Chapter 3

Biological vision

Mammals sense the world in many ways; among humans vision is used for daily tasks ranging from survival skills (e.g. feeding, defence) to contemporary needs such as crossing streets, reading, watching films, etc. The machinery behind this rich behaviour, the nervous system, is complex and simulating/emulating such a system on digital computers and in real time is a difficult task. Taking inspiration from biology has proven to be a good way to produce energy-efficient, real-time visual systems; to this end we need to study the visual pathway and its components. The principles described in this chapter have been adopted to develop the networks presented in Chapters 5 and 6.

The visual pathway

The sensory and processing elements responsible for vision are usually known as the *visual pathway*. They comprise the eye, the *lateral geniculate nucleus* (LGN) and multiple zones of the *cortex* (Figure 3.1).

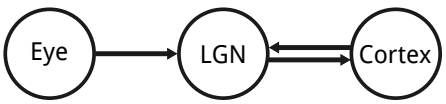
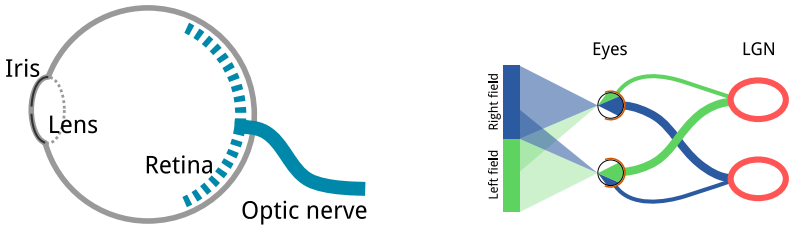


Figure 3.1: General connectivity of the mammalian visual pathway.

At the front-end of the visual pathway the eyes sense light and pre-process the image of the world. Inside each eye there is a wall of neurons known as the *retina* (Fig. 3.2a) which is dedicated to converting light into events representing the state of the visual world. Interestingly, most of these events are generated when noticeable



(a) (b)

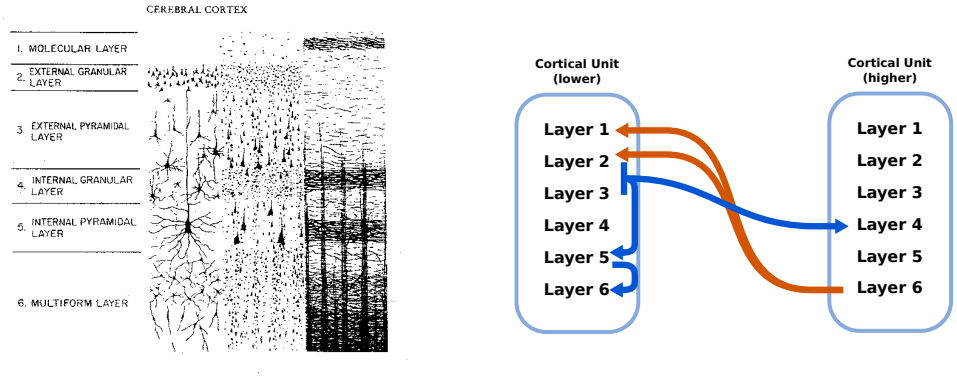
Figure 3.2: The front and middle stages of the visual pathway: a) The eye and its main components; b) Rewiring towards LGN assigns the right and left fields of view to a corresponding region of LGN.

changes are perceived. At the back of the retina, photoreceptors convert light into electrical signals which are then processed by a combination of neuron types (bipolar, amacrine and ganglion) to represent the image in the eye at multiple resolutions and to extract features (edges, motion).

The function of the Lateral Geniculate Nucleus is still an active field of research. From anatomy we know axons coming from the left and right eye are sorted before LGN (Figure 3.2b); while relay cells in LGN also sort and retransmit signals from the eye to the cortex [Hubel et al., 1995]. New research shows LGN is responsible for arbitrating signal transmission from the eye to the visual cortex, and that this part of the brain also receives feedback from the cortex; in fact there are more axons reaching the LGN from the cortical regions than the retina. Brainstem input to the LGN accounts for about 30% of incoming connections, which may also play a role in attention mechanisms [Ghodrati et al., 2017].

About 19% of the neurons in the brain are located inside the cortex which amounts to 80% of the brain's mass; this thin sheet of about 1100 cm^2 area and a 2 to 4mm thickness is responsible for the high-level cognitive tasks humans can perform [Herculano-Houzel, 2009; Thompson, 2000]. The fact that the cortex is thoroughly wrinkled accommodates a larger area sheet, and thus more neurons, to fit in the same volume. Moreover, regions which have a role in vision occupy around 30% of the cortex.

Cell distribution in the cortex seems to follow a pattern, as shown in Figure 3.3a. In a cross section, layers can be observed which are thought to have special functions, in particular we can classify layers by the direction of information flow (Figure 3.3b): the feed-forward path includes layer 4 which receives input from lower regions of



(a) (b)

Figure 3.3: Diagram of cortical cell types, distribution of cells, and external connectivity. a) Cortical column layers using different staining methods (Golgi, Nissl and Weigert) [Brodal, 1981; Kandel et al., 2000]. b) Intra- and inter-columnar connectivity.

the brain, and layers 2 and 3 whose outputs go to higher cortical areas [Hubel et al., 1995; Thompson, 2000]. On the feed-back path layer 5 takes input from the output layers and sends axons to layer 6, which in turn connects to layers 1, 2 and 3 in lower brain regions. Connectivity seems to be mostly distance dependent which leads to specialization of small regions known as cortical micro-columns, these are thought to be the basic building block of cortical function.

The eye

The retina is a complex sensory organ dedicated to encode visual information from the world into spikes. It has a layered structure (Figure 3.4); three of these layers contain neuron bodies, while two more have axons and dendrites. Light passes through the retina and is captured by *photoreceptors* to get transformed into electrical signals. There are two classes of photoreceptor: *rods* sense mostly achromatic illumination while *cones* have colour-selective response. Photoreceptors transduce light into a single type of neurotransmitter (glutamate), which, in turn, either excites or inhibits bipolar cells. This gives rise to two different channels for visual information transmission: the OFF channel which encodes decrement in illumination; while the activity of the ON channel reflects luminance increments [Hubel et al., 1995].

The next neurons in the retina are called *horizontal cells* and are thought to handle

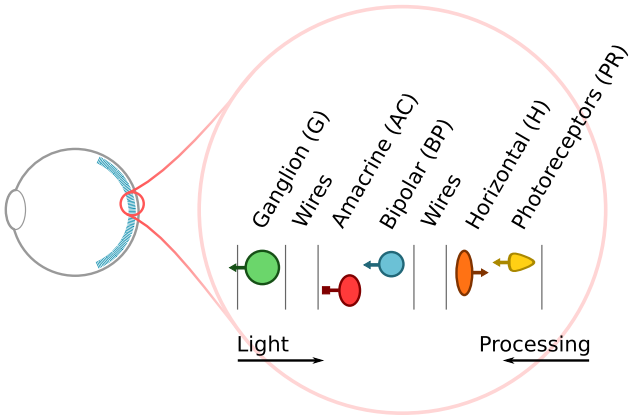
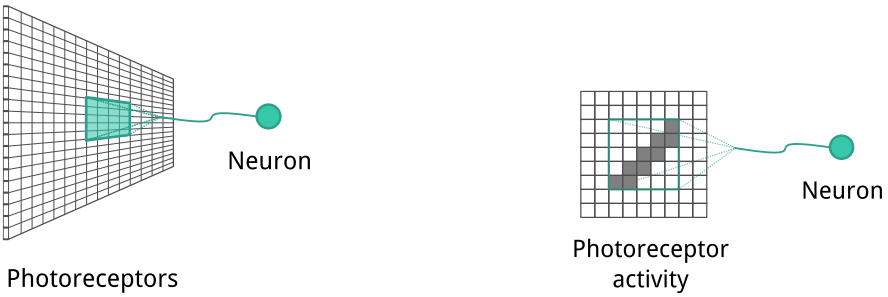


Figure 3.4: Layered organization in the retina



(a) (b)

Figure 3.5: Receptive field components: a) the sensory region which provides input to a neuron; b) in some instances the "shape" of the region is also important.

gain control in neighbouring photoreceptors, this would allow perception of approximately the same local contrast even when global illumination conditions have changed (e.g. being able to read in low-light conditions)[Masland, 2012]. Researchers have theorized that bipolar, amacrine and ganglion cells apply a series of transformations so that a more efficient representation is sent out to the brain through the spiking behaviour of *ganglion cells*.

Each ganglion cell has a particular set of photoreceptors which, indirectly, feed its inputs (Figure 3.5). This set, or the input activity pattern, is known as the neuron's *receptive field* (RF).

In 1953, Kuffler analysed the retina as a black box; a stimulus was presented to the eye and retinal ganglion cells' axon responses were recorded. The conclusion was

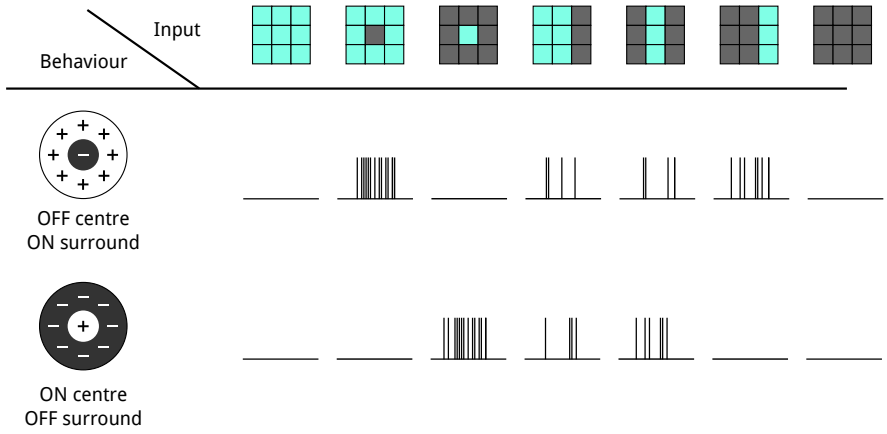
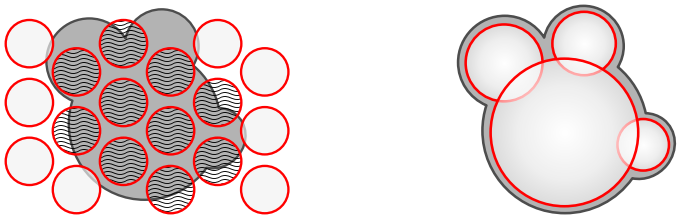


Figure 3.6: Neuron response to centre-surround receptive fields. When the whole receptive field is stimulated, no response is expected. If the input matches the centre-surround shape, then it will have the highest response. If it is partially stimulated, then the neuron will have a low response and will probably be inhibited by other neurons' activity.

that the receptive fields of ganglion cells are organized with a centre-surround motif (left of Figure 3.6)[Kuffler, 1953]. In essence the ganglion cell shows a high firing rate when there is activity in its receptive field's centre but almost none in the surrounding areas. This can reduce the activity sent out of the retina, for example an evenly lit object would only trigger some ganglion cells, particularly on its edges.

New activity recording and anatomical data acquisition methods have shown specific responses from retinal neurons prior to ganglion cells; this has lead to the proposal that bipolar neurons are prime computational units in the retina. It has been shown that bipolar cells have peculiar connectivity patterns in their input so that some respond to oriented bars or colour patches, in addition to typical centre-surround behaviour [Euler et al., 2014]. A combination of bipolar, motion sensitive neurons (starburst amacrine cells, SAC), and ganglion cells allow for direction selective response in the retina [Borst and Euler, 2011]. Figure 3.7b shows a cartoon of how having a variety of receptive field sizes (white circles) could lead to sending fewer spikes (four, one per RF) when compared to a uniform size sampling. The latter is illustrated as wave-filled circles in Figure 3.7a, where a total of 10 neurons are required to encode the input pattern.

Since energy is a scarce resource for animals it makes sense that sensors would aim to produce an efficient encoding of their inputs which reduces the number of active output neurons [Field, 1994; Thorpe et al., 2001]. Furthermore, has been shown that most



(a) (b)

Figure 3.7: Receptive fields of different size: a) uniform size sampling – 10 neurons would be required; b) multiple size sampling – 4 neurons required.

of the energy consumed by in the SpiNNaker machine is used in the computation of synapses Stromatias et al. [2013]. Scientists have proposed that ganglion cells with overlapping receptive fields compete to reduce spike emission and, thus, energy consumption [Bhattacharya and Furber, 2010]. Furthermore, by performing competition, only the ganglion cell which best describes the image patch should fire. An additional benefit of competing neurons is that, if ganglion cells' receptive fields encode similar information, redundancy is reduced while maintaining encoding robustness.

A way to implement competition is using feed-forward inhibition with the help of inhibitory inter-neurons [Thorpe et al., 2001]. Figure 3.8 shows an example of competition. We assume the receptive field of a couple of ganglion neurons receive a diagonal line. One of the bipolar cells (blue, top-left) has input connections whose spatial pattern matches the input while the other (blue, bottom-left) has an orthogonal one (squares on the left of bipolar neurons). This will lead to high activity[1] in the top bipolar, in turn it will make its paired inhibitory inter-neuron become active. By crossing the targets of inter-neurons, the competition mechanism is formed; in the example, the top ganglion wins and the bottom one is inhibited.

A basic model for retinal function is a single scale, centre-surround (via image convolution) operation. This is usually expressed as a Laplacian of Gaussian (LoG) filter, which is commonly carried out by applying Gaussian smoothing followed by the Laplacian operation.

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left(1-\frac{x^2+y^2}{2\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.1}$$

[1] Illustrated here as events, though in the retina this computation is carried via voltage differences.

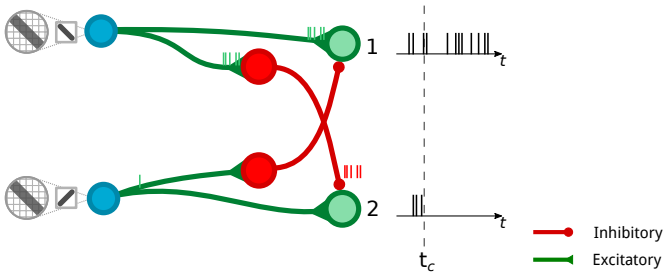


Figure 3.8: Competition between ganglions cells (green–right) executed using feed-forward inhibition through inter-neurons (red–middle), which are driven by activity coming from bipolar neurons (blue–left). Vertical bars indicate incoming activity to neurons. Output for the lower ganglion is reduced –even stopped– after t_c since a large inhibitory burst just arrived.

As observed in mammalian eyes at the ganglion level, when the input image is uniform, most of the output will be zeros; with the exception of edges. This model is used throughout the multi-scale network presented in Section 5.2.

Van Rullen and Thorpe extend this basic model to multiple scales of centre-surround receptive fields corresponding to ganglion cells of different *classes* (8 pairs of ON- and OFF-centre) [Van Rullen and Thorpe, 2001]. Instead of an LoG operation, they approximate the centre-surround filter via Difference of Gaussian (DoG).

$$g_c(x,y) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{x^2+y^2}{2\sigma_c^2}} \tag{3.2}$$

$$g_s(x,y) = \frac{1}{\sqrt{2\pi\sigma_s^2}} e^{-\frac{x^2+y^2}{2\sigma_s^2}} \tag{3.3}$$

$$DoG(x,y) = \mp g_c(x,y) \pm g_s(x,y) \tag{3.4}$$

Computing the spike times for each neuron in all scales makes use of *Matching Pursuit* [Durka, 2007] algorithm; the final representation is a rank-order code.

Using a similar approach, Bhattacharya and Furber created an algorithm to generate a sparse, rank-ordered spike representation of images [Bhattacharya and Furber, 2010]. This algorithm adopts the biological principle of competition and makes use of 2 ON- and 2 OFF-centre, each with a different size of receptive field. Even though this reduces the number of simulated cells, most of the image information is present with only 10% of the computed spikes. The output of this encoding is a sparse-distributed representation of the input, plus the order of the firings is the sole requirement to decode the image as it reflects how fit a ganglion cell is to encode a region of the image.

The team behind the *Virtual Retina* created a model which mimics both the cellular measurements and functional capabilities of retinal circuitry [Wohrer and Kornprobst, 2009]. Since it is a highly detailed simulation, it requires large computational resources – 100,000 neurons run at about $1/100$ of real time. Among other results this model allows for contrast gain control, spatial centre-surround filtering, temporal band-pass filtering, and foveated encoding.

Linear-nonlinear (LN) models have also been used to describe functional aspects of retinal ganglion cells, the principal characteristic of this family is the use of a linear combination of the receptive field activity which, in turn, is the input to a non-linear function; additionally, the firing behaviour of ganglion cells is stochastic [Karklin and Simoncelli, 2011; Pillow et al., 2005; Schwartz and Rieke, 2011]. These models have also successfully replicated cortical cell responses and been linked to efficient coding principles, though these studies do not explicitly produce a sparse representation.

A deep convolutional neural network (CNN) was trained to replicate the behaviour of a salamander retina [McIntosh et al., 2016]. The authors recorded ganglion cells when the retina was presented with natural images, subsequently they took measurements to train a 3-layered CNN. While the network matches the behaviour of the recorded ganglion cells, it is unclear whether these are representative of the full retina and the function of each cell type is still an open question.

We believe the retina is compressing the image with an over-complete set of basis vectors and sending this representation to the next stage where it can be processed and learned. In summary we can consider the retina as a feature-extracting computer; these features can be sent to later stages through spikes which should carry sufficient salient information for the cortex to adjust its perception of the visual world.

The Lateral Geniculate Nucleus (LGN)

Retinal axons carry visual information to the cerebral cortex, but before arriving at their final destination they reach a brain region known as the *Thalamus*; which is responsible for transferring signals from sensory organs into the cortex. Another function of the thalamus is to modulate information flow to and from the cortex; this could imply that signals from one cortical area could alter the behaviour of other areas [Sherman, 2006].

Since the Lateral Geniculate Nucleus (LGN) is part of the Thalamus it shares similar functions; in fact most research assumes it to be a simple relay station for signals coming from the retina. However, studies show that around 50% of the incoming

connections to LGN originate in the cortex, this indicates it is not just a relay but a modulator. Perhaps it could be a crucial factor for attention mechanisms observed in mammals. In the work presented in this Thesis we use connectivity principles from the LGN to reduce the number of spikes which would get sent to the plastic regions of our networks. The reduction is done, again, via competition where signals coming from the ON and OFF retinal channels inhibit each other if they represent similar inputs.

Similar to the organization in the retina, the LGN has separate regions for different scales of receptive fields comprising six "main" layers; two low- and four high-resolution versions of the image. In addition to these layers, the LGN has inter-layers which receive other image representations; scientists suggest these include motion, orientation and colour information measured at the retina [Hubel et al., 1995]. Furthermore, projections from both eyes reach the LGN in alternating layers, perhaps this organ is also responsible for a rough image registration mechanism.

Usually, LGN modelling is omitted from biologically-plausible visual pipelines as we do not require relays in computational versions of the visual pathway.

Funke et al. proposed different connectivity patterns for incoming retinal projections which may explain orientation preference in V1, the beginning of the visual cortex [Funke et al., 2002].

Sterling follows recent studies on LGN synapses to model a "quasi-secure" synapse, which promotes energy reduction, since the main result is fewer spikes emitted while information per event is incremented [Sterling, 2015]. Furthermore, the author suggests LGN is quite likely to be responsible for preparing neuronal groups to reduce long axons towards V1.

The LGN was modelled as a neurons with a set of Difference of Gaussian (DoG) receptive fields by Azzopardi and Petkov, the output of these filters are presented to a set of cortical simple cells. The main result from this is it achieves closer reconstruction of contour shapes than the ones obtained from using Gabor filters as receptive fields for simple cells [Azzopardi and Petkov, 2012]. The authors, however, do not consider image processing at the retinal level.

Computational models of the visual cortex

Much notable work in neuroscience has been driven by the study of visual areas of the cortex. In early studies researchers identified areas in the brain whose activity correlated with visual stimuli, these areas are known as the visual cortex. In 1963 Hubel

and Wiesel discovered the basis for most models of the visual cortex; they established a hierarchy of regions, a columnar organization of these regions in which every column has a higher reaction to a particular input. Within columns, they noted a particular connectivity pattern and established the concept of *simple* and *complex* cells. Simple cells receive the direct input from a previous region (either cortical or thalamic), while complex cells receive input only from simple cells.

Cortical architecture

While the simple/complex concept is still applied, it is not considered entirely correct as recent statistics on neural connectivity show that simple cells receive the *majority* of their input from previous regions but not exclusively; a similar correction is required for complex cells [Funke et al., 2002].

Fukushima developed the *Neocognitron*, one of the first neural networks to model the visual pathway [Fukushima, 1988]. The units in the network receive and output non-negative (rectified) analogue values; the input to the network is a simple receptor layer. This model has a hierarchical organization, layers after the input are composed of simple and complex cells. Features are learned and, later, extracted by simple cells; the role of complex cells is to introduce some translation invariance and provide the output for the current layer. Input connectivity is distance dependent, thus simple cells are exposed to only a portion of the previous layer. Weight updates are Hebbian inspired as they depend on local information only, though the network requires layer-wise training.

The architecture of the *HMAX* model was inspired by studies done on the visual cortex. It consists of two layers of alternating simple and complex cells; the former sample their input from previous layers and the latter do a MAX operation on the output of simple cells [Riesenhuber and Poggio, 1999; Serre et al., 2005]. At the top of the network a classifier is added to simulate the Infero-Temporal (IT) portion of the visual cortex. In the most recent version of the algorithm, there is some learning done in the simple cells of the second layer, but training is done off-line by sampling different corner-like patterns from the dataset.

In the first layer simple cells are tuned to oriented bars (S1 in Fig. 3.9) while at the second layer, cells respond more strongly to complex features (e.g. corners, crosses, longer lines). Complex cells in each layer provide invariance to small translations and rotations. This model has been successful in replicating some of the feed-forward behaviour of the visual pathway, though it does not include any lateral or feed-back

paths.

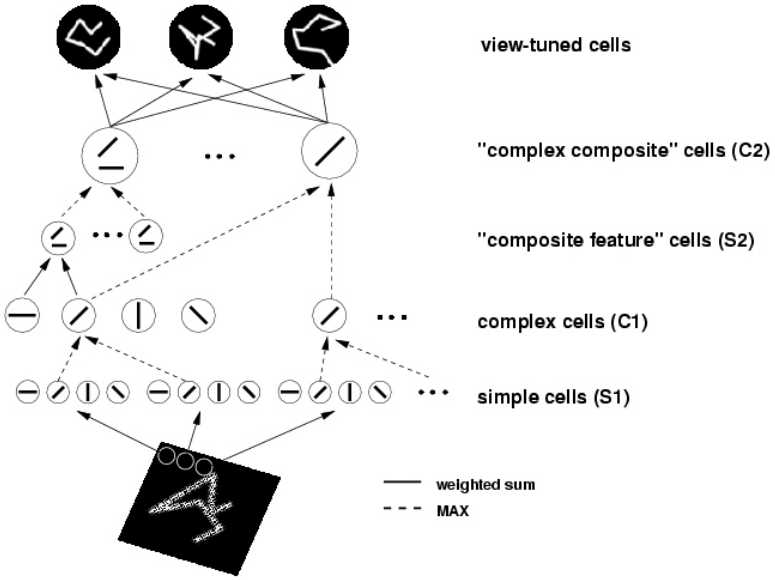


Figure 3.9: HMAX model[a]
[a]From http://maxlab.neuro.georgetown.edu/images/hmax

LeCun et al. devised an artificial neural network based on the connectivity of the visual cortex described by Hubel and Wiesel. In this model simple cells perform a convolution of the inputs while complex cells do a pooling operation [LeCun et al., 1995], this is not far from the HMAX network architecture though training of weights is applied throughout the entire network. Weights for simple cell stages are stored as convolution kernels, this brings layer-wise weight sharing and, most importantly, parameter reduction. As in the HMAX model, pooling provides certain invariance to local translation.

Training is done through the *Back-Propagation* algorithm [LeCun et al., 1989], which uses the chain rule to propagate the error measured in the output layer (right-most in Fig. 3.10) to intermediate layers. In this way convolution kernels get adjusted to better fit the statistics of the presented dataset. It is unlikely that in biology this algorithm is doing synaptic plasticity, since it requires high precision error propagation.

In 2003, Behnke presented a model, the *Neural Abstraction Pyramid* (NAP), which presents invariance to multiple image transformations [Behnke, 2003], as is believed to happen in the visual cortex. Similarly to HMAX, the NAP divides each layer into simple and complex cells; the former neurons receive and integrate inputs, meanwhile complex units output the computation results. The addition of feed-back and lateral

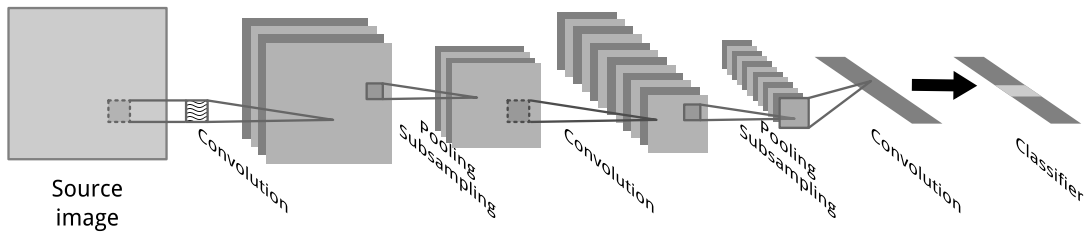


Figure 3.10: Cartoon of a Convolutional Neural Network.

signals allow the network to spread higher layer representations into lower layers, the abstraction grows as layer depth increases. Lateral projections enable correlations between large areas to be taken into account in the learning process.

The *Hierarchical Temporal Memory* (HTM) model is also based on the architecture of the visual cortex. It is similar to other models in that there is topographical correspondence in receptive fields and as the hierarchy level increases concepts get more abstract. Ahmad and Hawkins take another concept from biology as inputs get changed into sparse, distributed representations which provide robustness and generalization properties [Ahmad and Hawkins, 2015].

Another distinct characteristic of HTM is the use of time as a supervisor [Hawkins and George, 2006] so that the output from each layer is affected not only by spatial information but also by previous inputs. An HTM network (Figure 3.11) is composed of cells, which are packed into columns; collections of columns are known as regions which have the same receptive fields, and many regions form layers. The main data paths are the feed-forward, carrying current input signals to higher layers, and the feed-back which brings predictions to lower layers; lateral connectivity also carries predictions to the same layer cells.

An interesting property is that, as we go up the hierarchy, temporal information changes more slowly than in previous layers (left of Figure 3.11). At the lowest layer, columns sense small areas, with simple and fast-changing patterns; at the top, each column should sense a wide receptive field, with complex and slow-changing patterns. Statistics on the input is captured by the formation of synapses between cells; to form a synapse an active source cell needs to send a signal to a target column. If there is a cell in an active or predictive state, a synapse is formed or strengthened, otherwise it gets weakened.

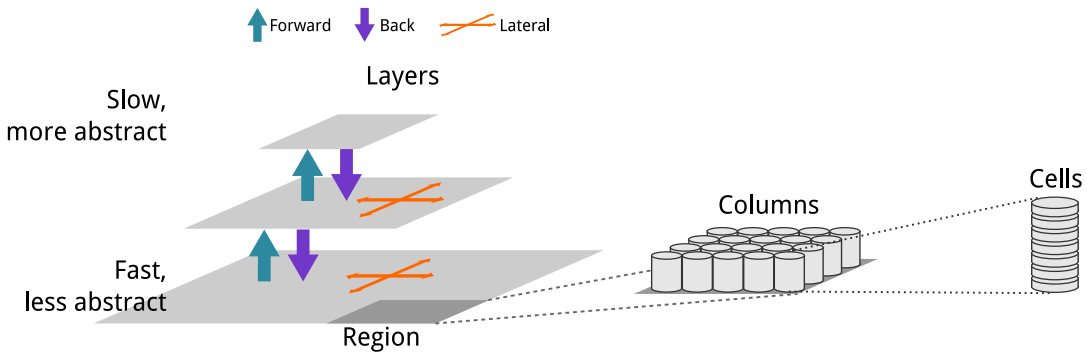


Figure 3.11: Cartoon of an HTM Network.

Summary

One of the principles of neural processing seems to be information per signal maximization, this could ensure energy reduction.

Heavy pre-processing of input data is prevalent in sensory organs; for example, the cochlea in the ears transforms sounds to a frequency representation. Something similar occurs for the retina where different sizes of receptive fields transform the input into a spatial-frequency representation.

Neurons in V1 area of the cortex have been shown to respond to contours even under occlusion. Hierarchical processing is a common motif through the cortex, distance dependence also seems prevalent (regions of visual or tactile input or frequency in audio). A hierarchy of processing elements allows for more flexible function modelling while maintaining element count relatively low.

This connectivity is sparse and localized, thus some optimizations can be used in the SpiNNaker Toolchain. In the next chapter a network expander is presented which is executed on SpiNNaker cores (as opposed to on-host) which improves network loading times.

Chapter 4

Topographic connectivity deployment

Particular zones in the brain seem to be responsible for the movement of different muscles; moreover, nearby activity in the cortex incited muscle movement in close by regions [Gross, 2007]. In the case of vision, signals coming from the eyes reach zones in the back region of the brain; these zones are organized in a topographic, hierarchical fashion [Hubel and Wiesel, 1963]. We also follow such organization so that neurons in our networks are connected to its inputs with a distance dependence. Using specialized hardware to simulate realistic neural circuits typically requires uploading a network model; this process, if not optimized, could take longer than the simulation itself. Furthermore, some of the networks in Chapter 5 have been modelled as convolutional layers (with repeated weights) whose loading onto SpiNNaker can be heavily optimized. The conventional way to upload a network can be summarized in two steps: first, generate an explicit representation on the host machine (a list of each synapse present in the network) and, second, send it to SpiNNaker. Instead of this procedure, we send a high-level description of the connectivity (e.g. distance dependence or connection probability) to the SpiNNaker machine which, in turn, generates the explicit connectivity. The goal of this chapter is to produce a set of algorithms which run on the SpiNNaker machine and generate biologically inspired (distance-dependent, image kernel) neural network connectivity. By reducing the amount of data transferred we achieve faster loading times; however, in some cases, the capabilities of the SpiNNaker cores proves to be a speed limitation. Details of the efficient connectivity deployment method are described in this chapter.

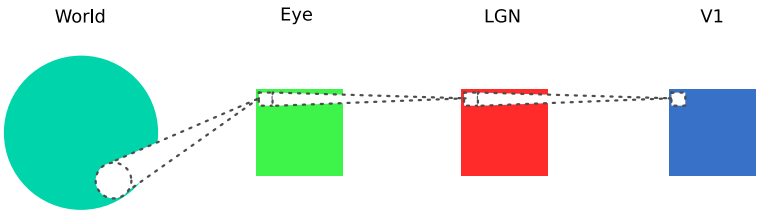


Figure 4.1: Regions which sense a particular area of the visual field tend to be located together.

Topographic connectivity in the visual pathway

The topographic correspondence between the visual field and cellular location in the visual pathway is known as *retinotopy* (Fig. 4.1). There is evidence that later regions in the visual pathway retain topographic connectivity (although abstraction increases) [Fox et al., 1987; Grill-Spector and Malach, 2004]. Measurements taken using modern recording techniques of brain activity (e.g. Functional Magnetic Resonance Imaging [fMRI], tomography) have allowed researchers to confirm this spatial relationship [Tootell and Hadjikhani, 2001].

Topographic connectivity can be achieved through distance dependence; if we assign two-dimensional (2D) coordinates to each pre- and post-synaptic neuron, a synapse can be formed only if the distance between neurons is smaller than a certain value. For static networks it is useful to express weights and delays in the form of matrices, or kernels. To simulate the different functions of the proposed retina and LGN models (Chap. 5) we require to generate static, distance-dependent connectivity. To deploy these networks, we created a kernel-based connector for the PyNN [Davison et al., 2009] spiking neural network description language. In the case of the cortical model, the required connectivity is dependent on both distance and probability, which are present in PyNN but as separate entities; thus we created a "Cortical" connector to combine these dependencies.

Image kernel-based connectivity

High-level descriptions of networks are decomposed into SpiNNaker-compatible graphs. To generate proper connectivity we have to filter which nodes of the graph should be connected. In general, it is a process of trimming-down the edges of an All-To-All connectivity into a specialized type.

For ths notis image kernel-based connectivity we assume both pre- and post-synaptic populations are 2-dimensional (2D) grids and indices are row-major mapped. Transformations between 2D and 1D representations are done by the following equations:

$$\begin{aligned} index &= (row \times width) + column \ , \\ row &= \lfloor index/width \rfloor \ , \\ column &= index \bmod width \ ; \end{aligned} \tag{4.1}$$

where $width$ is the width of the 2D grid and $u \bmod v$ is the remainder of the division of u by v.

We may want to connect two populations which are themselves output to kernel-based inputs. Spatial sampling is not necessarily done one-to-one, thus we require sufficient information to transform both populations into a common space. Transforming original coordinates (both rows and columns) to and from sub-sampled ones:

$$\begin{aligned} sampled &= \left\lfloor \frac{original - offset - 1}{step} \right\rfloor + 1 \ , \\ original &= offset + (sampled \times step) \ ; \end{aligned} \tag{4.2}$$

where $offset$ is the distance from the origin and $step$ is the spatial sampling frequency (stride) the connector uses.

Once the coordinates of neurons are transformed into a common space, we can measure the distance between them and discard those which are too far away. Since we make use of a convolution kernel of known dimension the weights for these connections are set by the kernel, we discard synapses from a pre-synaptic neuron if the latter is outside the rectangle defined by

$$\begin{aligned} K_{area} = \Bigg[&\left(row_{post} - \frac{kernel_{height}}{2}, \ column_{post} - \frac{kernel_{width}}{2} \right) ; \\ &\left(row_{post} + \frac{kernel_{height}}{2}, \ column_{post} + \frac{kernel_{width}}{2} \right) \Bigg] \end{aligned} \tag{4.3}$$

Figure 4.2 shows the connectivity generated between source and target populations. The source represents a 2D grid of 6 rows and 8 columns, while the target has 3 rows and 4 columns. The weight kernel has a width and height of 5 pixels, the offset and sampling rate are set to 2 pixels.

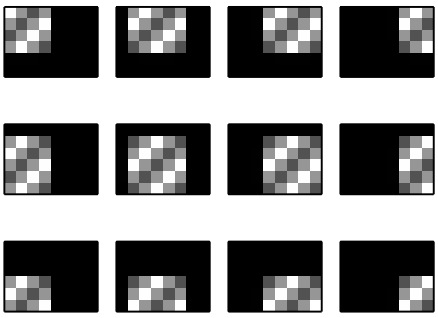


Figure 4.2: Generated connections using a weight kernel. Each rectangle represents a post-synaptic neuron from a population with 12 neurons (a 4×3 resolution). Pre-synaptic neurons are represented as a pixel in each of the rectangles; the pre-synaptic population represents an image of a 8×6 resolution. The weight kernel used had a 5×5 which is shown as non-black pixels in each rectangle.

Cortical connectivity

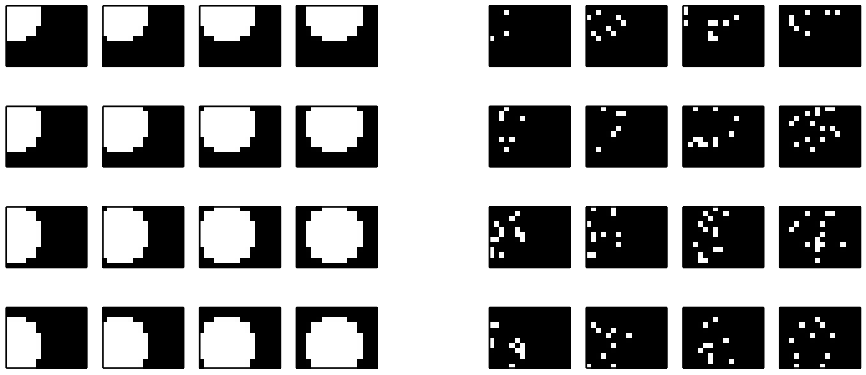
Our cortical connector is similar to the Kernel-based one except that, even if the pre-synaptic neuron lies within the sampling area of the post-synaptic one, we randomly choose whether to generate a connection or not, and do so using fixed, uniform probability. Although we still use a row-major mapping for population indices, each coordinate in the 2D grids can hold multiple neurons (sub-population). This leads to a small change to the set of equations 4.1, for which we need to define a *total width* to take into account these sub-populations:

$$width_{total} = width * N_{pz} \,, \tag{4.4}$$

where N_{pz} is the number of neurons sharing the same coordinate.

$$\begin{aligned} index &= row \times width_{total} + column \times N_{pz} + index_{sub} \\ row &= \lfloor index / width_{total} \rfloor \\ column &= \lfloor (index \bmod width_{total}) / N_{pz} \rfloor \\ index_{sub} &= index - (row \times width_{total} + column \times N_{pz}) \end{aligned} \tag{4.5}$$

If the distance between pre and post neurons is shorter than the designated radius (i.e. sampling zone; white regions in Fig.4.3a), we generate a random sample to decide whether a synapse should be formed between the neurons.



(a) (b)

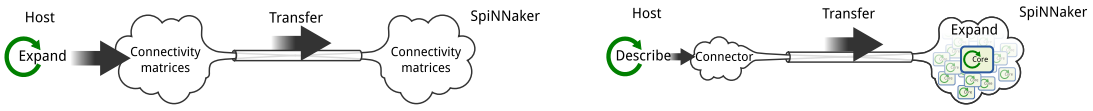
Figure 4.3: Cortical connector. a) Sampling region per target neuron, each square is a post-synaptic sub-population. White circles are the pre neurons which are taken into consideration to generate synapses. b) Generated connections, zoomed at top-left, typical cortical connection probability is 10%. Every square represents a post neuron, every white dot a connection with a pre neuron.

Figure 4.3b show connections generated between the pre- and post-synaptic neurons. Each rectangle represents a post-synaptic neuron and generated connection are depicted as a white dots. These dots also represent the position of the pre-synaptic neuron.

Efficient neural network deployment on the SpiNNaker machine

While these new connectors do not produce dense matrices, when the input size increases, the time taken to load the network can be comparable to, or even larger than, simulation time. The main reason for this is that the current SpiNNaker software tool-chain (SST) requires connectivity matrices to be transferred as fully detailed lists, where each item must contain: source neuron id, target neuron id, weight and delay. To reduce the time to load, we can change from an explicit to a high-level representation. In the following sections we describe a neural network expansion system for the SpiNNaker machine.

There are two main components to the problem of network-loading times (Figure 4.4a): first, the computation of network connectivity matrices has to be done serially on host; and second, these matrices have to be transferred without compression.



(a) (b)

Figure 4.4: Generating connectivity on host vs on SpiNNaker

The time-to-load problem can be diminished by transferring high-level descriptions of the connectivity, instead of the full connectivity matrices, and expanding these descriptions into connectivity matrices in parallel on SpiNNaker (Figure 4.4b). To implement this we modified the SST in three aspects:

- The SST was altered to collect connectivity descriptions and prevent on-host expansion,
- A Python interface was built to control execution of SpiNNaker binaries, and
- Code for SpiNNaker was developed to receive and expand descriptions.

Description collection

The SST takes a spiking neural network described by neuron *Populations* which are connected through *Projections*. Each Projection, in turn, contains a description of the connectivity through a *Connector*[1]. The SST contains a *Synaptic Manager*[2,3] object whose purpose is to take Connector's descriptions and transform them into many <pre-, post-synaptic index, weight, delay> tuples.

Since we are going to generate weights and delays on SpiNNaker, the main changes were applied to the Synaptic Manager object, to prevent the SST generating connectivity matrices if Connector objects allow on-SpiNNaker expansion. In the definition for this object, we later collect minimal descriptions for Connector objects as described in Table 4.1

Since we want to reduce the amount of data transferred, Names are changed from pure text (ASCII) into a cyclic redundancy check (CRC) hash [Koopman, 2002]. Additionally, some of the required parameters are compacted to the minimum functional size, e.g. boolean values are represented by 8-bit variables.

[1] /path/to/sPyNNaker/spynnaker/pyNN/models/neural_projections/connectors
[2] https://github.com/chanokin/sPyNNaker
[3] /path/to/sPyNNaker/spynnaker/pyNN/models/neuron/synaptic_manager.py

Table 4.1: Minimal connectivity description.

Data	Required information	Requirement example
Source vertex	Portion of the source neuron population	start and count
Target vertex	Portion of the target neuron population	start and count
Connector type	Name and requirements	probability of connection
Weight type	Name and requirements	values range
Delay type	Name and requirements	constant value
Synapse type	Name and requirements	plasticity parameters

Additionally we require to compute matrix sizes and generate a data structure, called *population table*, to map pre-synaptic (source) population <key, mask> pairs to synaptic matrices memory addresses. We store address starts, row lengths, source key, source mask and synapse types (excitatory, inhibitory, modulatory, etc.) in simple lists. Each entry in these arrays corresponds to a pre-synaptic population key and mask which allows us to track where a connector expander should start writing data.

Table 4.2 shows the information contained in a population table entry. To match population table entries we traverse each row from which we extract the key and mask elements and match them to the address start information. Once this data is collected, it is sent to SpiNNaker so that it can generate connectivity information on its own.

Table 4.2: Population table entry.

Name	Description
Key	Pre-synaptic population identifier
Mask	Pre-synaptic neuron filter
Start	Offset to retrieve data from matrices address array
Count	Number of entries in the matrices address array

Connector expander

Python interface

The SpiNNaker-based connector expansion must be controlled from the Host, so that we know when it starts, finishes and if it runs into problems. For this purpose we developed a Python interface[4,5] whose main functions are to search for cores which will simulate target populations and upload to SpiNNaker the expansion executor. If the connectivity includes delays greater than the ones supported by default (16 simulation time steps), we upload another binary to the appropriate *delay extension* cores. This Python launcher gets called by the main SST Python interface[6] after the connectivity descriptions have been uploaded to SpiNNaker.

SpiNNaker executor

For the connector expansion to be executed, we created an application which is responsible for generating source/target neuron id pairs, weights and delay values. For each source neuron id we create a *synaptic row* as described in Table 4.3.

Each entry can be of one of two types: *static*, stored in 32-bit entries, its data does not change and whose format is described in Table 4.4; or *plastic*, where weights change but control data does not (unless structural plasticity is present [Bogdan et al., 2018]) and its format is similar to the static entry, except weights and control are split into two separate regions. A "plastic-static" region is identical to the lower 16 bits of a static entry where we specify target neuron id, synapse function (excitatory, inhibitory, etc.) and delay. The second region can be composed of one or more 16-bit variables, one of which must be the weight.

When the on-SpiNNaker expander detects larger-than-standard delays in connectivity entries, it sends a delay/target-id pair to delay extension cores. These pairs are processed and added to a record of which target neurons will require a delayed relay. The record is a collection of a bit-arrays, in this case arrays of multiple 32-bit words whose bits have a particular meaning. In this case, the bit position is equal to the presynaptic neuron id and its value tells us whether this neuron is expected to be delayed a certain amount of simulation time.

[4] https://github.com/chanokin/SpiNNFrontEndCommon

[5] /path/to/SpiNNFrontEndCommon/spinn_front_end_common/connection_builder

[6] /path/to/SpiNNFrontEndCommon/spinn_front_end_common/interface/abstract_spinnaker_base.py

Table 4.3: Synaptic row format.

Row	Contents
-1	number of indirect elements (30 bits) | tag (2 bits)
0	number of plastic elements (N)
h	plastic headers (could be empty)
1	second plastic weight | first plastic weight (16 bits)
$\vdots$	$\vdots$
$N = \lceil n/2 \rceil + h$	last plastic row | penultimate plastic row
$N+1$	number of fixed elements (F)
$N+2$	plastic control elements (P)
$N+3$ / F_1	first fixed entry
$\vdots$	$\vdots$
$N+F+2$ / F_F	last fixed row
$N+F+3$ / $P_{1,2}$	second plastic control | first plastic control (16 bits)
$N+F+\lceil P/2 \rceil+2$ / $P_{P-1,P}$	last plastic control | penultimate plastic control

Table 4.4: Synaptic connection entry.

	Weight	Delay	Type	Target
Bit width	16	4	x	8

Each bit-array in the record is known as a delay block and will be assigned a temporal delay range of multiples of the maximum standard delay (Figure 4.5), and so the spike will be kept in the relay until the appropriate delay has passed. Since standard delays are applied at the target population, there is no need for higher temporal resolution per block.

Benchmarks

To test parameter generation we used the values of synaptic weights, for which we used constant and probabilistic generators; Figure 4.6 shows histograms of the created values on SpiNNaker (left, green) and on Host (right, blue). In each experiment we created a neural network consisting of single source and target populations with 5000 neurons each. We connected them through a one-to-one connection which gives us

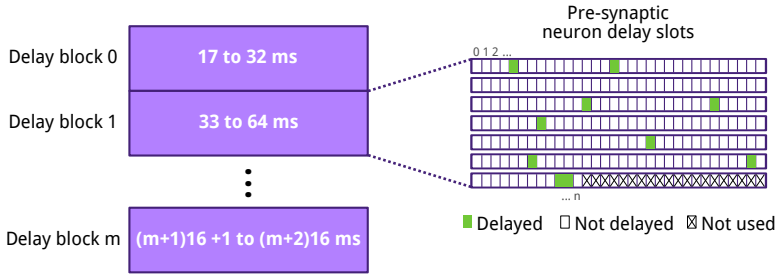
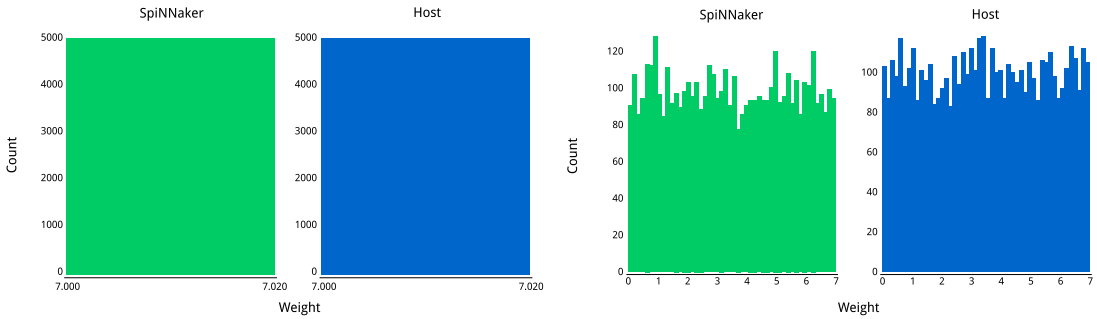


Figure 4.5: Delay extension blocks.

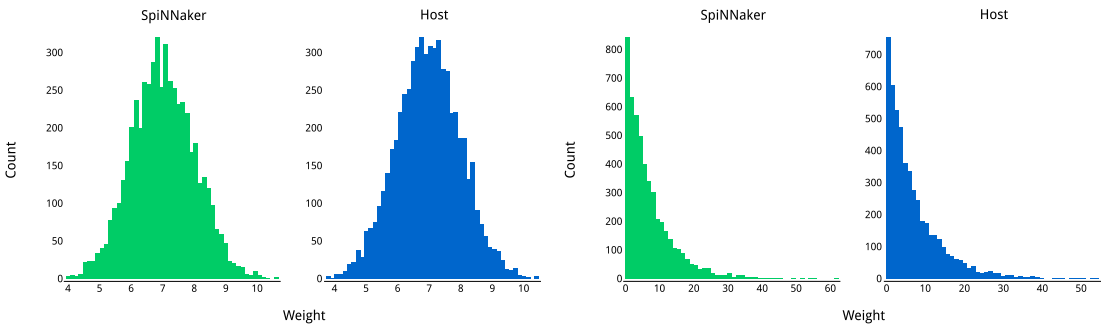
5000 samples for the generated parameter (weights). For each of the available parameter generators we ran simulations with generation on Host and on SpiNNaker. The number and value for weights created by the constant (Fig. 4.6a) and probabilistic generators (Figures 4.6b, c, and d) have similar distributions in all cases. In the case of the constant generation the target value was 7. For the uniform generation the range was chosen to go from 0 (inclusive) to 7 (exclusive). The mean and standard deviation where 7 and 1, respectively, for the normal distribution generator. The exponential distribution was generated with the parameter $\lambda = 0.1$.



(a) Constant (b) Uniform

Figure 4.6: Parameter generation on SpiNNaker vs. Host

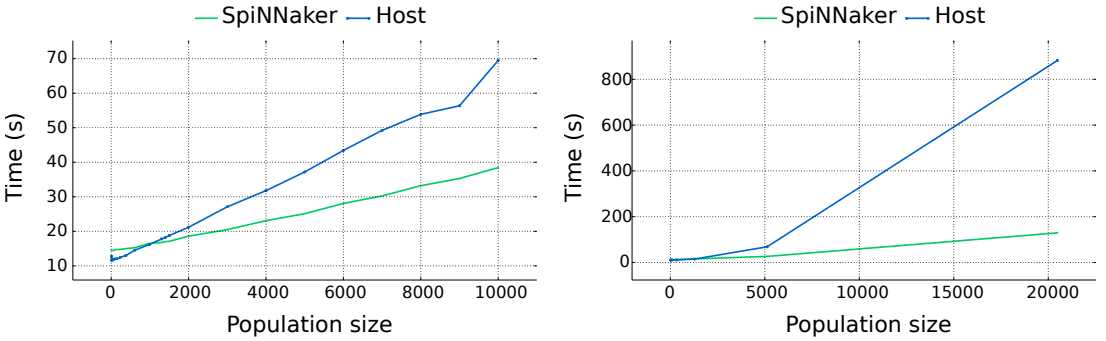
The main bottleneck with Host-generated connectivity was transfer time. To test if any load-time reduction was achieved using on-SpiNNaker generation, we measured and compared the time it takes to run a short simulation (10ms). For these experiments we set source and target populations of different sizes (8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1300, 1400, 1500, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000), with static constant weight values, and increasing delay values (1, 17, 33); we measured the total execution time of each experiment and averaged over combinations for each population size.



(c) Normal (d) Exponential

Figure 4.6: Parameter generation on SpiNNaker vs. Host (cont.)

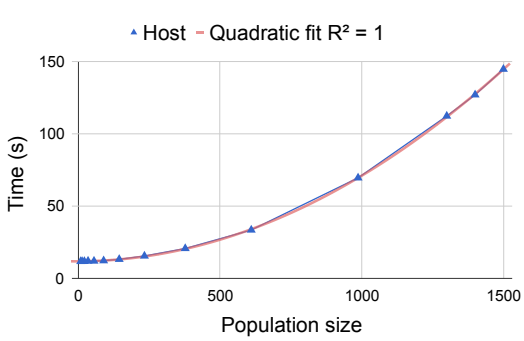
In Figure 4.7a we observe that for the one-to-one connectivity the growth in both cases is linear, though the rate for on-SpiNNaker expansion time is lower. We did a similar test for the Kernel-based connector, where the sizes of both source and target populations changed from (10, 8) to (160, 128), at each step we multiply dimensions by 2. Similarly, the growth rate in the case of expanding the network on SpiNNaker is much smaller.



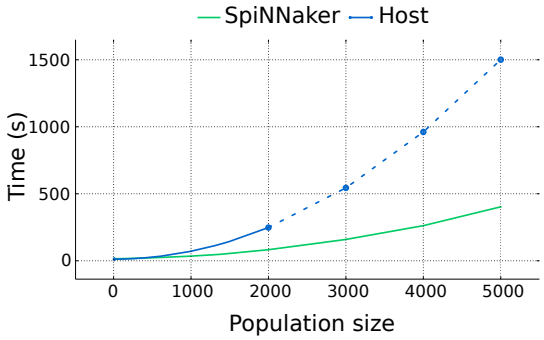
(a) One-to-one (b) Kernel-based

Figure 4.7: Synaptic matrix generation time comparison

For the all-to-all connector case, we stopped testing at 1500 neurons per population since weights started saturating on-host RAM plus transferring weights took multiple hours. To estimate loading times for higher population sizes (dashed portion of the blue curve in Figure 4.8b), we did polynomial curve fitting (Figure 4.8a) which resulted in a quadratic fit. Figure 4.8b shows a simulation time reduction for each real comparison (populations with 1500 or less neurons); furthermore, it shows dramatic

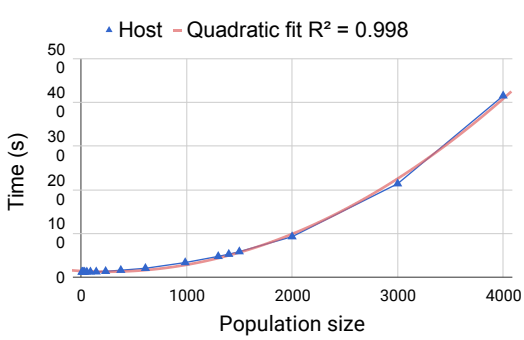


(a) All-to-all curve fit

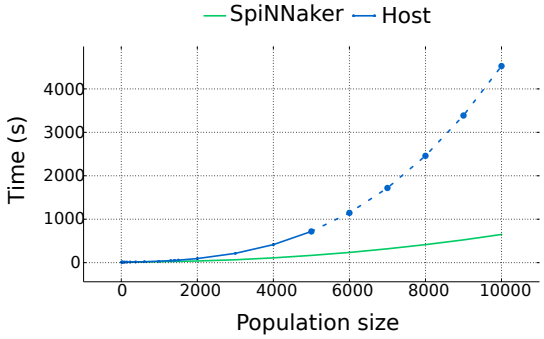


(b) All-to-all

Figure 4.8: Synaptic matrix generation time comparison (cont.)



(a) Fixed-probability curve fit



(b) Fixed-probability

Figure 4.9: Synaptic matrix generation time comparison (cont.)

speed increment for the (estimated) large populations when compared with the on-SpiNNaker network expansion.

The last standard connector we tested is the fixed-probability one. Similarly to the all-to-all case we were unable to test for more than 4000 neurons per population when connectivity was generated on host. We performed curve fitting for the host-based experiments which results in a quadratic polynomial as shown in Figure 4.8a. In Figure 4.9b we see less performance gain than the all-to-all case, most likely due to the slow random number generation on SpiNNaker and low connection probability (10 %).

A test for the Cortical connector was not performed since it is similar to the Kernel connector, this is each post-synaptic neuron connects to a fixed number of pre-synaptic neurons. The main difference would be that the SpiNNaker random number generator would slow down the generation.

Summary

We have shown how changing an explicit representation into a high-level description can be used as an efficient method to upload statistics-based neural connectivity. In small networks we deliver a comparable load time, though on-Host generation can still be faster. For large networks, the difference can be perceived as waiting minutes instead of hours. Expansion of kernel-based connectors is an important step towards using spiking convolutional networks on SpiNNaker.

In the next chapter input pre-processing networks are discussed; these make extensive usage of the connector expander and the custom connectors developed

Chapter 5

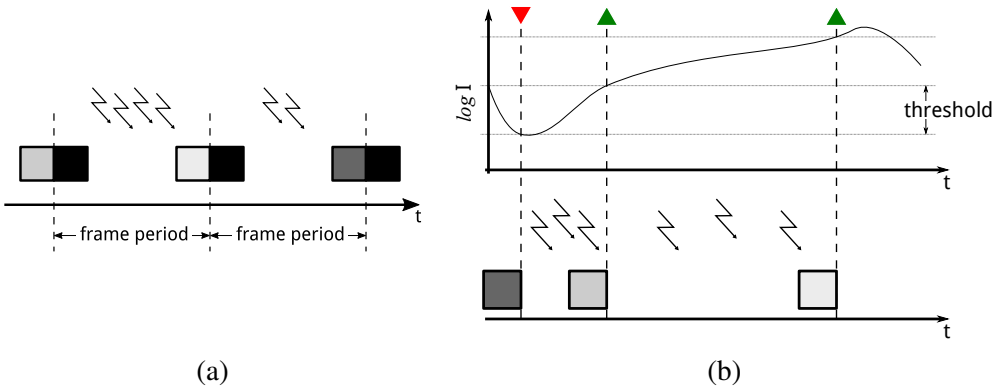
Sensing the visual world

Any visual system requires an input sensor, in this Chapter we present a couple methods in which we can transform images (or video) into spiking representations. Traditionally computer vision has used frame-based sensors which capture light for a fixed period and store it. While this approach works for full image reproduction, it may not be the only (or optimal) way to provide input to computer vision algorithms.

In nature, organisms have to optimize resources; an important aspect is to transmit information from sensors using as few signals as possible, usually through cells called ganglions [Field, 1994; Rao and Ballard, 1999; Srinivasan et al., 1982]. This chapter will present networks inspired by biological retinas which further reduce spike counts by changing the image representation. Furthermore, we demonstrate a spiking version of the Reichardt motion detector [Borst and Euler, 2011] and oriented bar detection networks. These networks could serve as building blocks for a computer vision pipeline and have served as the basis of some networks presented in Chapter 6.

Image conversion to spikes

The simplest way to transform traditional images is to assume that a sensor does nothing but sense light, converts it to spikes and transmits these signals to the visual pipeline. This can be achieved by simply grabbing every pixel of an image and encoding its value into spike rate (e.g. 0 to 255Hz). A common way in the field is to use rate code (Section 2.3.1) using Poisson processes [Liu et al., 2016]. Since the range of the spike rate can be arbitrarily chosen, this conversion may have high bandwidth requirements.



(a) (b)

Figure 5.1: Comparison of visual sensors: a) Camera photoreceptor behaviour; light is accumulated for a fixed time period, the value is sent out of the sensor for further processing. Finally the pixel's value is reset. b) Neuromorphic vision sensor photoreceptor behaviour. The sensor emits events whenever the log light intensity changes above a certain threshold.

Neuromorphic vision sensor emulation

It is commonly thought that eyes work as cameras; there are however several differences, most significantly, that ganglion cells activate when they sense a sufficient change in brightness. This differs significantly from traditional image sensors composition since pixels in digital cameras accumulate light for a fixed period of time and then output values in what is known as a *frame* (Figure 5.1a); if the camera is shooting video, this process is repeated at a fixed rate.

Computer vision researchers have traditionally used conventional cameras whose frames require updating regardless of changes in the scene; furthermore, most techniques require processing this full image which is energy expensive. Neuromorphic Vision Sensors (NVSs) [Berner et al., 2013; Mead and Mahowald, 1988; Pardo et al., 2015; Serrano-Gotarredona and Linares-Barranco, 2013] adopt principles from biology to reduce energy consumption while keeping fast response times and great dynamic range. Their pixels accumulate light, if the change in the log-light-intensity is greater than a threshold they emit an event (shown as triangles at the top of Figure 5.1b). Since events are driven by changes in the scene, if the latter remains static, no events are generated. This is closely related to how the neural circuitry in the eye works although, in biology, multiple receptive field sizes reduce output rate.

Extensive work has been done to generate multiple computer vision datasets (e.g.

MNIST, CIFAR, YouTube8m) [Abu-El-Haija et al., 2016; Krizhevsky et al., 2014; LeCun et al., 2010]. Since SpiNNaker's "native" language is spikes, we were interested in converting such well known databases into an event-based representation, but without passing through a noisy channel (i.e. computer screen to NVS). Moreover, by controlling the conversion process we are able to introduce deterministic motion (similar to micro-saccades), noise and customize the output encoding.

To utilize these large group of datasets and to approximate how mammals perceive imagery we propose to convert images (or video) into a spike representation. Our conversion mechanism [Pineda García et al., 2016] works by functionally emulating a hardware NVS, the main difference is that, in our software, discrete sampling steps are used. We assume images are measurements of initial stages of eye circuitry which can then be compared to a reference value to decide if an output signal would be emitted. The basic approach involves computing the difference D between the input image I and the reference level matrix R (Eq. 5.1), then compute an update mask matrix M as shown in Equation 5.2.

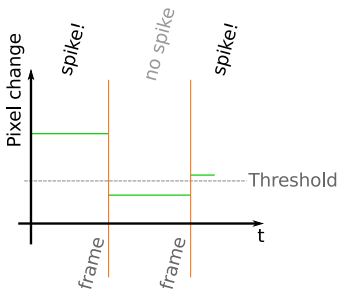
$$\mathbf{D}_t = \mathbf{I}_t - \mathbf{R}_{t-1} \tag{5.1}$$

$$\mathbf{M}_t = f(\mathbf{D}_t, \mathbf{H}) = \begin{cases} 0 & |d_{t,i,j}| < h_{t,i,j} \\ 1 & |d_{t,i,j}| \geq h_{t,i,j} \end{cases} \tag{5.2}$$

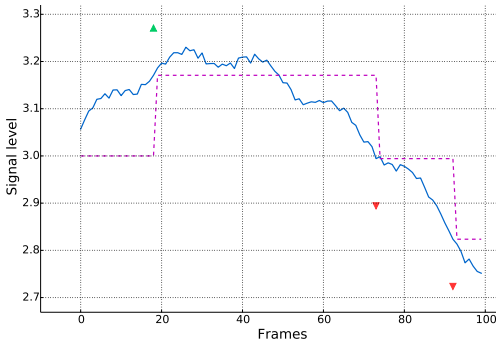
$$\mathbf{R}_t = \mathbf{R}_{t-1} + \mathbf{M}_t \odot \mathbf{D}_t \; ; \tag{5.3}$$

where $\mathbf{R}$ is initialized to zero ($\mathbf{R}_0 = \mathbf{0}$) and $\mathbf{H}$ is a per-pixel threshold matrix. In Equation 5.2 $d_{t,i,j}$ stands for the value of the difference between input and reference for pixel $< i,\ j >$ at time t; in the same equation $h_{t,i,j}$ means the state of the (adaptive) threshold for pixel $< i,\ j >$ at time t. Finally, the reference level can be updated with 'valid' differences, $\mathbf{M}_t \odot\ \mathbf{D}_t$ (Equation 5.3, Figure 5.2a), where the $\odot$ operator indicates element-wise multiplication.

We compared the NVS emulator, using different thresholds, to the behaviour of hardware reported by Serrano-Gotarredona and Linares-Barranco [Serrano-Gotarredona and Linares-Barranco, 2013]. As the baseline for this experiment we recorded the activity coming from the NVS when exposed to a translating black-and-white image. At the same time, we pointed a digital camera (PS3-Eye [Sony Corp.]) to the same screen and passed its output to the NVS emulator. Finally, we generated a sequence which corresponded to the translating image, a virtual camera, and provided this to the NVS emulator. These measurements were repeated for the NVS emulator with different



(a)



(b)

Figure 5.2: Default NVS model behaviour: a) When the difference between the current pixel value and its corresponding reference is larger than a threshold, an event (spike) is generated. b) Photoreceptor model. Whenever the reference (magenta-dashed) and signal (blue-continuous) are separated by at least a threshold (5% $\approx$ 0.16) a spike is produced (green-up and red-down triangles).

threshold values (5, 10 and 20 % of the brightness scale).

We counted generated events per frame period ($\sim$16ms) for each experiment and normalized them. Figure 5.3a shows the behaviour when the threshold was set at 5% (12/255 brightness). The hardware NVS and the virtual-camera exhibit peaks and valleys which correspond to the image being on the screen and off, respectively. The digital camera shows spikes of activity, these are a product of the automatic gain and white balance mechanisms.

These spikes diminish as we increase the threshold value (Figure 5.3b) and, finally, disappear when the threshold reaches 20% (Figure 5.3b). This is not an optimal way to reduce the activity peaks (Figure 5.3), as we are basically making the algorithm less sensitive and it is losing information. An alternative may be to use adaptive thresholds to reduce spike counts when these large bursts of light are perceived.

We want to make the emulator to have some resilience to noisy pixels or dropped spikes; to achieve this we enhanced it with biologically-plausibility in mind (Figure 5.4a). Firstly, we switched the receiver model from a perfect one (i.e. never dropping an output spike) to a receiver which takes into account possible spike drops. If we think of the NVS emulator as an emitter and further stages in the visual pathway as receivers, it is desirable to think of an unreliable transmission line and to model this aspect the receiver model was changed to a ‘forgetful’ one. Since the reference value is constantly being diminished the same input level will still generate events in this

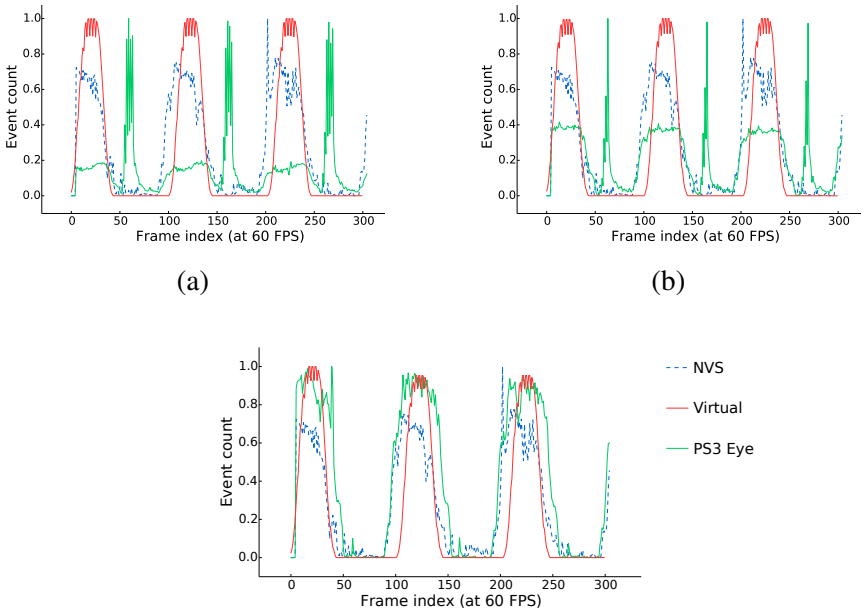


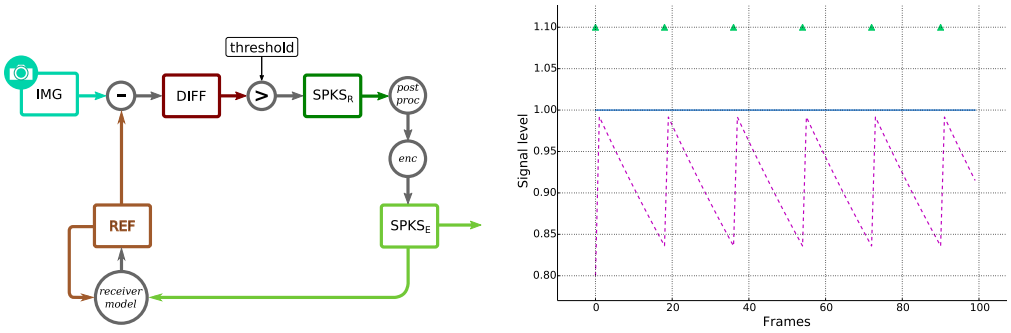
Figure 5.3: Behavioural comparison given thresholds on NVS emulator. Different thresholds were used a) 5%, b) 10%, c) 20%; in every plot blue-dashed, green and red lines indicate hardware NVS, digital camera + emulator, virtual camera + emulator, respectively

way; if some events are lost in the transmission line, the receiver would still be able to recover to a correct value after some time, Equation 5.3 will be changed to

$$\mathbf{R}_{t+1} = w_f \mathbf{R}_t + \mathbf{M}_t \odot \mathbf{D}_t \ , \tag{5.4}$$

where $0 \leq w_f \leq 1$ is the rate at which the reference level value approximates to its default value; we can observe the behaviour of the system given a step input in Figure 5.4b.

Adapting the emulator's behaviour to the input signal rate of change would allow it to gain sensitivity and detect slow-changing input. Additionally, it could diminish the output rate when the input signal constantly alternates between high and low values (e.g. noise, a faulty pixel). We require to change the *constant* threshold matrix $\mathbf{H}$ into a spike-dependent value,



(a) (b)

Figure 5.4: Enhancements to basic photoreceptor model. a) Enhanced NVS model diagram. Threshold, spike post-processing and a noisy receiver model where added to the basic model. b) Reference level behaviour to step input, when the threshold equals $5\% \approx 0.05$ and decay rate $w_f = 0.99$.

$$\mathbf{H}_{t+1} = \mathbf{H}_t \odot g(\mathbf{D}_t, \mathbf{H}_t) \tag{5.5}$$

$$g(\mathbf{D}_t) = \begin{cases} w_{down} & |D_{t,i,j}| < H_{t,i,j} \\ w_{up} & |D_{t,i,j}| \geq H_{t,i,j} \end{cases}, \tag{5.6}$$

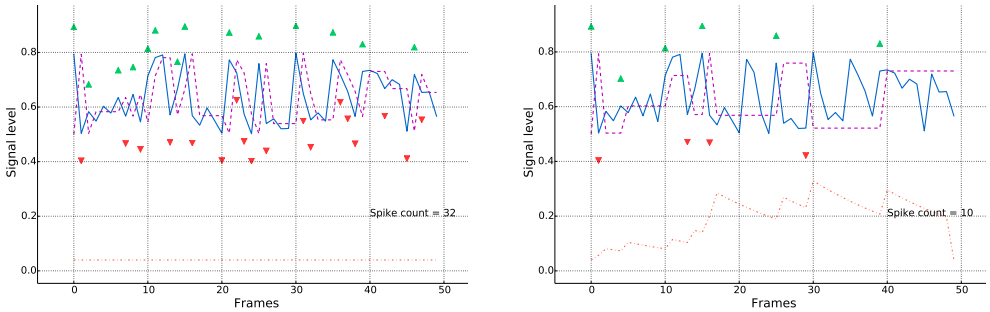
where $\{w_{down} \in \mathbb{R} \mid 0 < w_{down} < 1\}$ and $\{w_{up} \in \mathbb{R} \mid w_{up} > 1\}$.

We demonstrate the adaptation to fast-changing signals first; in Figure 5.5b we observe how the basic NVS emulator (i.e. constant threshold and non-decaying reference) behaves, in this particular case most changes in the signal will elicit an event; this could lead to many spikes being generated (Fig. 5.5a).

Another benefit of having an adaptive threshold is being able to sense slow-changing signals, Figure 5.6 shows such a signal. On the left we see a static threshold response; assuming the reference level is initially equal to signal, no events occur since the difference between time steps is never larger than the threshold.

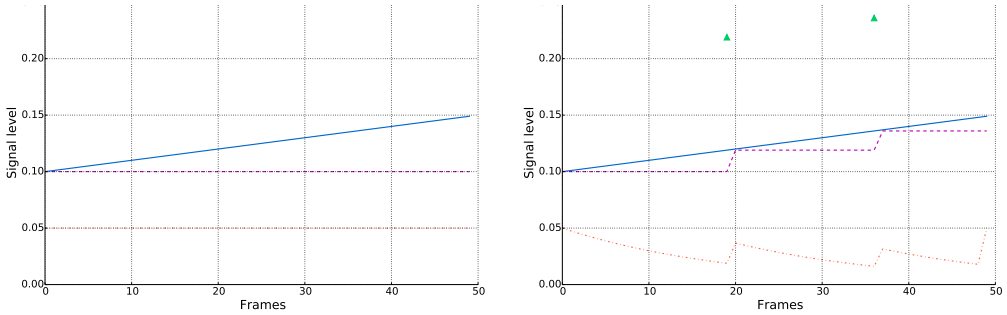
We can observe the benefits of having an adaptive (decreasing) threshold in Figure 5.6b. As the simulation progresses the threshold value drifts towards zero (or a minimum value); this opens the possibility for smaller changes to trigger an event which, in turn, increases the threshold again.

Finally, we test all the components of the NVS emulator model together (Fig. 5.7), by decaying the reference value and adapting the threshold. In Figure 5.7a we observe the system's response to a step input; the emulator will initially spike fast to catch up



(a) (b)

Figure 5.5: Comparison of threshold behaviours, fast-changing signal. Blue-solid line depicts the input signal; magenta-dashed line represents the NVS emulator reference value; the orange-dotted line at the bottom of the plot is the threshold value.



(a) (b)

Figure 5.6: Comparison of threshold behaviours, slow-changing signal. Blue-solid line depicts the input signal; magenta-dashed line represents the NVS emulator reference value; the orange-dotted line at the bottom of the plot is the threshold value.

with the input signal value until achieves an almost periodic output balanced by the rate at which it forgets its history and threshold adaptation. We now evaluate the response to noisy signals, when the noise is not large enough to cause spikes in every frame (Fig. 5.7b) the emulator behaves in a similar way to how it reacts to the step function; this means that by adapting the threshold the photoreceptor mostly ignores this noise while still generating spikes even if the average value does not change.

If we add noise which can cause spikes, the NVS emulator will respond mostly to large changes and ignore low-power noise. By adapting the threshold and decaying the reference level, we can filter out noisy sources while making sure a receiver could still recover from loss of communication.

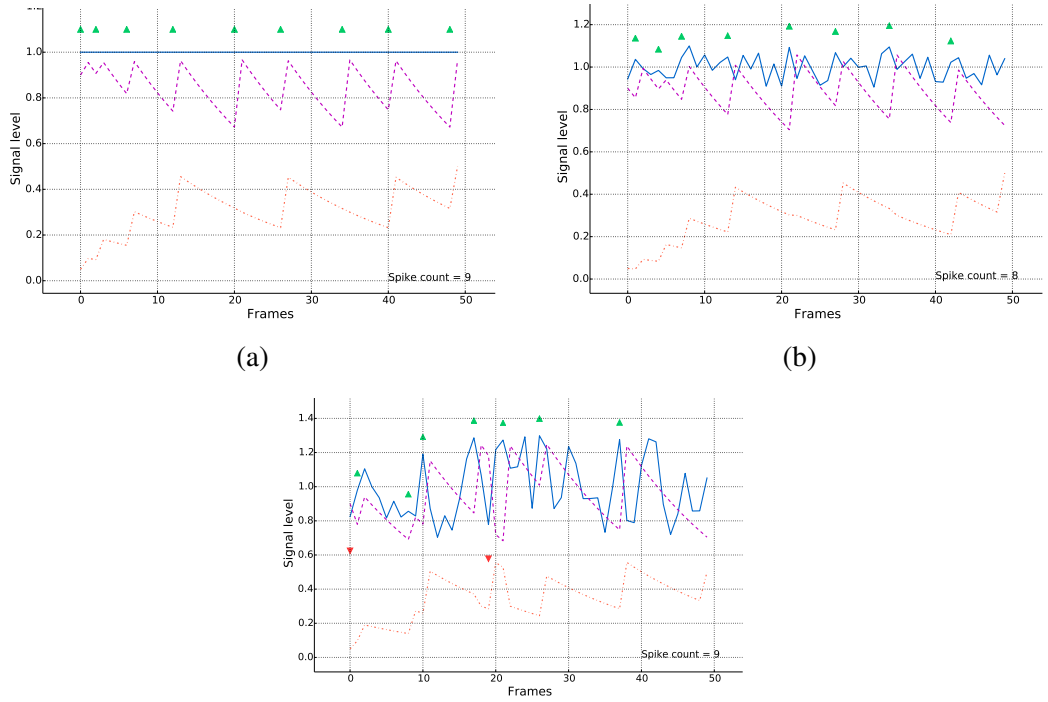


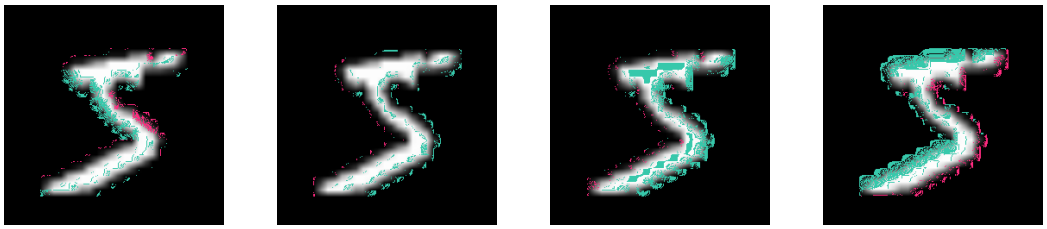
Figure 5.7: Response to signals when the NVS includes an adaptive threshold and history decay. Blue-solid line depicts the input signal; magenta-dashed line represents the NVS emulator reference value; the orange-dotted line at the bottom of the plot is the threshold value. a) Step function; b) small and fast changes; and c) large and fast changes.

.

Rank-ordered neuromorphic vision sensor

Evidence that different receptive field sizes are present in retinal ganglion cells has been known for decades [Hammond, 1974]. The previous NVS-based image encoder can be thought of as an eye model which takes into account ganglion cells with a single-size, high-resolution receptive field. This implies that a single version of the visual field is being captured, this could result in an unstable representation. Adding different receptive field sizes adds multiple scale representations of the same input, which can both stabilize the representation and may be used for different tasks, such as answering the questions what are we seeing and where is it in the field of view.

Furthermore, researchers have theorized that the cortex uses sparse, distributed representations (SDRs) to store and transmit information due to their capacity, responsiveness to similar inputs and robustness to noise [Field, 1994; George and Hawkins,



(a) (b) (c) (d)

Figure 5.8: Moving a scaled-up MNIST digit (from 28×28 to 256×256), and simulating a micro-saccadic movement (1 pixel per frame) at 90 frames per second. In cyan are pixels representing a brightness increment and magenta the ones for decrements.

2009; Kanerva, 1988]. To produce this representation the encoder would require to reduce redundant activity from cells which share receptive field regions and change of basis vectors from a single pixel to patches. If the basis vectors for the transformation are not orthogonal, reducing redundancy has the effect of pushing orthogonality to the new representation. A sensory encoder which transforms it's input to an SDR would be able to feed cortical regions with its "native" language.

We can achieve such a sparse visual sensor by changing the simple encoder in our NVS emulator to a rank-order one and adding an appropriate decoder (Figure 5.9). The rank-order encoding we have used here was developed by Bhattacharya and Furber [2010] and it uses four different sizes of centre-surround receptive fields. To simulate the cells with these receptive fields we perform a convolution on the input image with an image kernel The weights of the kernel are computed using a difference of Gaussians centred at the middle of the image and normalized so that it sums to zero and its auto-correlation equals to one. After the convolutions, we interpret each pixel as a neuron and their value as the order in which they would spike; each time a neuron spikes it reduces the value (order) neighbouring neurons. The algorithm stops when a required number of neurons have spiked, typically, around 10% of them.

To test whether we obtain a reduction in the number of spikes using such rearrangement of the NVS emulator we use a more complex (richer in textures) image to which we apply a simulation of micro-saccades. We define a micro-saccade as a random translation of at most 1 pixel in each direction (both horizontal and vertical) and we perform one micro-saccade per frame. After each micro-saccade we do a step of the rank-order NVS emulator and we send out 5% of the possible spikes.

Figure 5.10 shows the evolution of rank-order NVS: the left column depicts the current reference which is also the reconstruction the decoder block in Figure 5.9; the

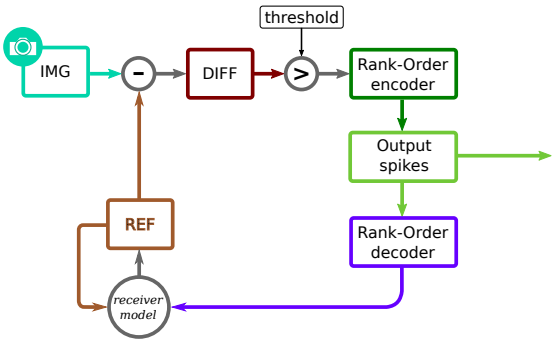


Figure 5.9: Diagram of a rank-order neuromorphic vision sensor. It includes a complex spike encoder for the reference, input difference and an appropriate decoder.

central column shows the difference between the reference frame and the current input; finally, the right column displays raster plots of outgoing (from ganglions) spikes. In each raster plot we organized neuron ids according to their receptive field size, this way bottom to top (1 to 4) show decreasing resolution.

In the first row of Figure 5.10a the reference for the rank-order NVS model is at its default level, thus most pixels will be "active" after the difference operation. Using the rank-order encoding we can appreciate that most activity is in two of the scales (2nd and 4th), the topmost corresponds to lowest resolution. After the first frame we can obtain a fairly good reconstruction, in particular of broad shapes, as seen in the middle row (Figure 5.10b) Sending the second batch of spikes further refines the reference, we now see the output transition from having the most activity in the low spatial frequency ganglions (layer 4) to the ones with the smallest receptive field (layer 1). Since the total number of output pixels is around 3 times the ones in the input image and we are sending 5% of that count in the output, we are constantly sending around 1.5% of the total input pixel count. This is reducing the number of spikes while increasing the information per spike and, by changing the image representation we could gain some robustness to small transformations (e.g. translation, rotation, scaling) in the input.

Event count dynamics

A more realistic environment to test how much reduction in spike counts would involve larger translations than a micro-saccade. We now assume there are two types of eye movements (implemented here as image translations): micro-saccades which have a small magnitude (≤ 1 pixel) and full saccades whose magnitude is usually much larger than a micro-saccade (tens of pixels). As in the previous section micro-saccades are

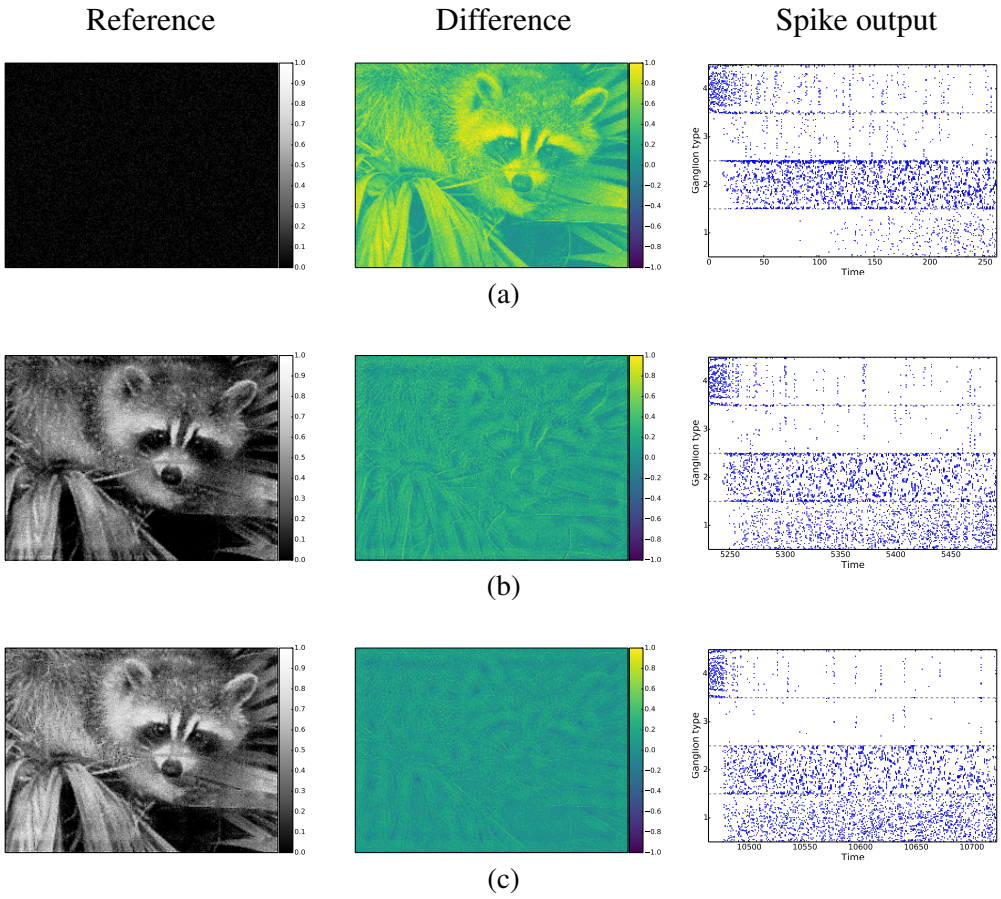
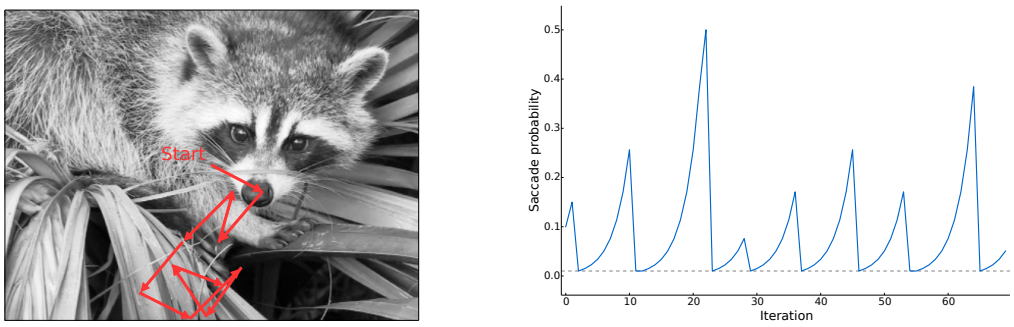


Figure 5.10: First three iterations of rank-order NVS. The left column shows the reference frame, the middle shows the difference between input and reference frames, and the rightmost column shows the raster plots of 5% spikes. In each raster plot spikes are ordered according to ganglion type of decreasing resolution as we go up in the plot.

generated at random and translate the image by, at most, 1 pixel; additionally, micro-saccades are applied to each frame unless a full saccade replaces it. The magnitude and direction of a full saccade are computed by subtracting the pixel with the largest difference (Eq. 5.1) and the centre of the image. The occurrence of full saccades is determined by probability,

$$p(saccade)_t = \begin{cases} p_{min} & \text{if saccade happened at } t-1 \\ \min(1, w_{up} \times p(saccade)_{t-1}) & \text{otherwise} \end{cases} \tag{5.7}$$

where w_{up} is a real number larger than one and it is used to increase the probability every frame; p_{min} is a constant which states the minimum probability. Every frame we



(a) (b)

Figure 5.11: Simple attention mechanism. a) sequence of saccades. b) saccade probabilities

sample a random value from a uniform distribution in the range [0, 1), if this value is lower than the saccade probability we randomly choose one of the five largest changes in the difference frame. In this experiment the values for saccade generation where $w_{up} = 1.5$ and $p_{min} = 0.01$; we send 10% of the generated spikes; and the forgetting rate of the NVS emulator reference is $w_f = 0.95$. Figure 5.11a shows the sequence of saccades simulated over the first 70 frames of a simulation and Figure 5.11b depicts the evolution of the probability of a saccade.

An interesting result of the experiment comes from analysing the number of spikes produced by each ganglion type (i.e. resolution), which is shown in Figure 5.12. At the beginning of the simulation most spikes come from the three lowest resolutions, led by ganglions with the largest receptive field (type 4); this likely happens due to the empty reference frame. After each saccade, the two middle resolutions have higher spike counts but decay while the lowest and highest resolution ganglions' rate increase. This is due to a better fit of middle resolution kernels to non-zero-valued pixels in the difference frame after a saccade.

Another nice property is that the number of spikes which represent high resolution steadily increase after the saccade and maintain activity. Since we are constantly moving the image and these changes are generally caused by edges, it is natural that these will activate small receptive fields. The lowest resolution ganglion type seems to follow a similar behaviour as the highest resolution one, this may be happening to maintain global luminosity in the image. These activity patterns and the fact that only 5 to 10% spikes are required for a good reconstruction indicate that it could be plausible to substitute a single-scale NVS by a multi-scaled one without incurring in much

bandwidth overhead.

Visual processing

The visual processing networks presented in this section take inspiration from the circuitry found in the mammalian eyes and are designed to use an NVS as an input. As such, we use a similar, common motif: bipolar cells (B) sample the input (P) with different image kernels and activate corresponding ganglion cells (G). Bipolar output also feeds accompanying amacrine cells (A) which, in turn, inhibit activation of neighbouring ganglion cells (Figure 5.13a). We chose to use the output of an NVS as it has an embedded ON–OFF channel separation plus it is already in SpiNNaker's "native" language.

In short, Bipolar cells detect features in the image, so their activity represents an image patch. As the features we detect are usually not orthogonal we require a mechanism to reduce redundancy in the image representation. We use amacrine cells, since they are of the inhibitory type and are fed by bipolars, to prevent spikes from ganglion cells whose activity represents a similar feature (i.e. image patches). As bipolars sample the input with an image kernel, we can compute how similar is the information which ganglions carry by calculating the cross-correlation of the input kernels. We maintain this connectivity motif on the different visual processing sub-systems.

General image representation

Inspired by the model described in Section 5.1.2, we propose a network for image representation based on different sizes of Gaussian receptive fields for bipolar cells and lateral competition on ganglion cells. We define different populations which scan

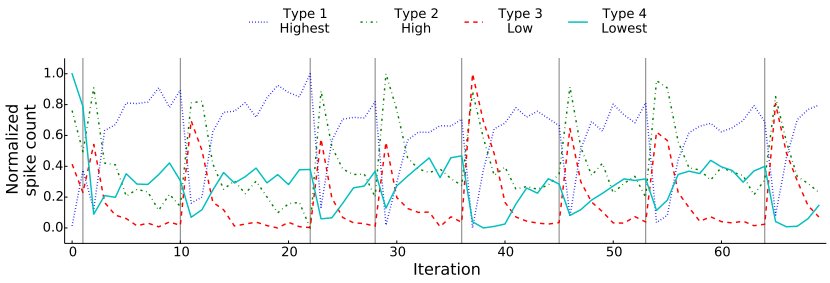
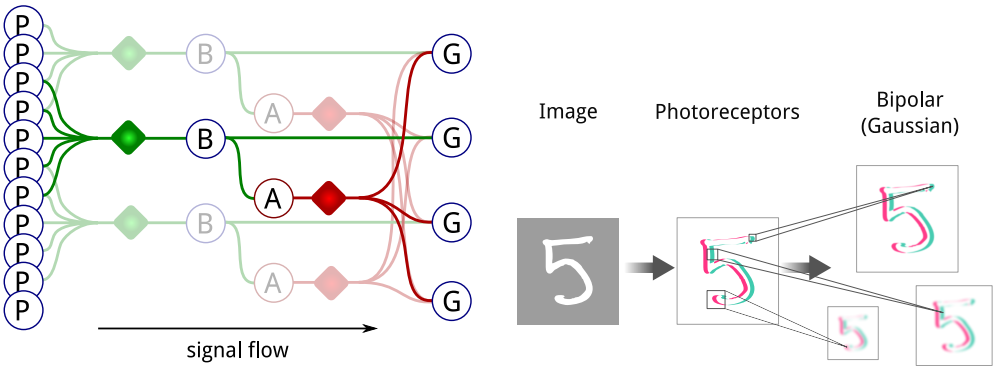


Figure 5.12: Spike counts for rank-order NVS with simple attention.



(a) Main connectivity motif diagram.

(b) Bipolar cells samples the image at different scales.

Figure 5.13: Aspects of visual processing SNN. Green lines indicate excitatory connections, red ones are inhibitory. Rhombus (green and red) indicate kernel-based connectivity. Bipolar cells sample NVS emulator spikes through a weight kernel, each bipolar excites a ganglion (output) and an amacrine (inhibitory inter-neuron). The latter inhibit neighbouring ganglions to reduce redundant activity.

the full input image with using weight (image) kernels whose values are proportional to 2D Gaussian functions; each neuron in a population shares the same input weights. Additionally, each population scans the input with strides proportional to the size of the weight kernel; for example, a kernel of width 11 produces the stride would be 3. These connectivity characteristics essentially reduce the scale of the input image. Since we have many populations using different weight kernels and strides, we end up with a multi-scale representation of the input image. This network can be thought of as an approximation to a matching pursuit algorithm (MPA) [Durka, 2007]. An MPA will take a set of over-complete basis to represent a signal via a compressed, sparse vector. In this case, the dictionary or basis are the bipolar input kernels and the spikes from ganglion cells represent the compressed vector. The reason this is not exactly an MPA is that no feedback signal will monitor the reconstruction quality. Since we require the same network to model different inputs, and its weights are not to be trained, we have to dynamically modify the interaction between neurons so that the function approximation property, to a certain extent, still holds.

In biology such compression is thought to arise from competition among neurons; there is evidence that such mechanism is present among retinal ganglion cells [Cook

and McReynolds, 1998; Portelli et al., 2016]. In our spiking network, competing neurons would send inhibitory signals to neighbours which adjust the input current by a certain factor,

$$I_{Total} = I_{ff} - I_{lat} \sim w_{ff} - w_{lat} \ . \tag{5.8}$$

The proposed connectivity motif generates centre-surround activity detected in *in-vivo* recordings of retinal ganglion cells. Inputs to bipolar cells are distance-dependent, whose weights are computed using a Gaussian distribution. Each Gaussian is centred at the bipolar coordinate ($\mu_x = x_c$ and $\mu_y = y_c$) projected back into the input space and have equal standard deviations ($\sigma_x = \sigma_y$),

$$G_i(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp^{(-[(x-\mu_x)^2/(2\sigma_x^2)+(y-\mu_y)^2/(2\sigma_y^2)])} \ . \tag{5.9}$$

To compute the inhibitory weights we calculate the cross-correlation between the two input Gaussian (G_0 and G_1) kernels [Bromley, 2017],

$$G_0(x,y) \star G_1(x,y) = G_0(x,y) \circledast G_1(x,y) = \frac{1}{2\pi(\sigma_0^2+\sigma_1^2)} \exp^{-(x^2+y^2)/[2(\sigma_0^2+\sigma_1^2)]} \ , \tag{5.10}$$

where $\star$ denotes the cross-correlation and $\circledast$ the convolution operator[1]. Note that if both deviations (σ_0 and σ_1) are equal, the new width of the area is $\sqrt{2}$ times the original, which would produce centre-surround receptive fields. Figure 5.14 illustrates the formation of this receptive field; in the top-left we can see the input kernel and the top-right figure illustrates the lateral inhibition kernel. Figure 5.14c shows that the subtraction of the input and lateral kernels would result in a centre-surround filter.

We set an experiment to test the behaviour of the centre-surround receptive fields in a SpiNNaker simulation; in this we use the highest resolution sampling. We converted MNIST digits into a spike-rate representation (Fig. 5.15a; Section 5.1.1) and feed it to the proposed network with three ganglion cell types which correspond to their receptive field width (Table 5.1).

To further reduce the likelihood of bipolar cells sampling similar features we chose a stride (how far apart they are placed) of roughly half of the receptive field diameter. The neurons in the network for multi-scale image representation use the parameters shown in Table 5.2

[1] We can use either since Gaussian functions are even.

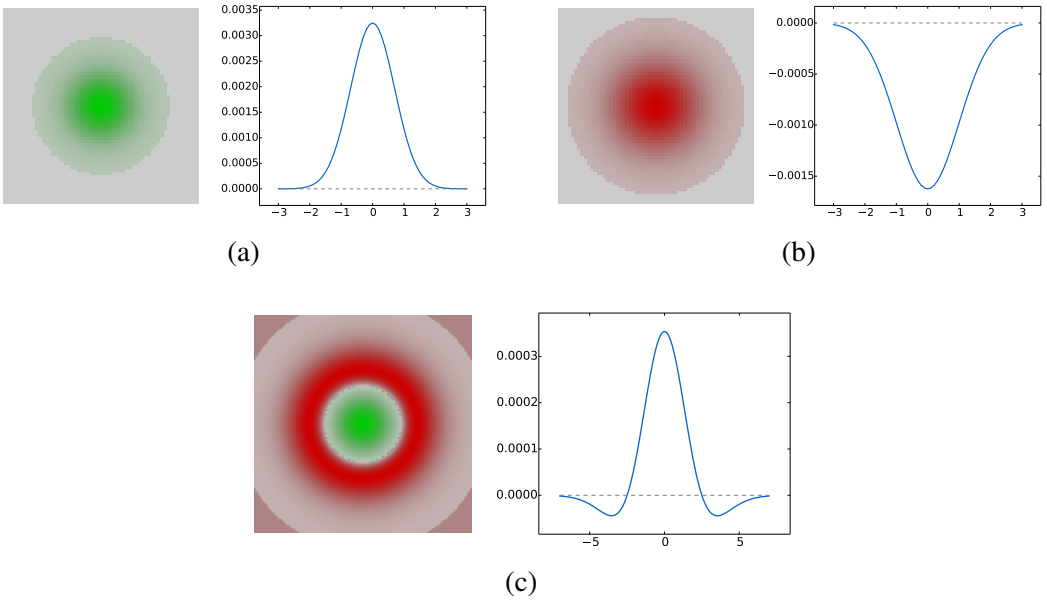


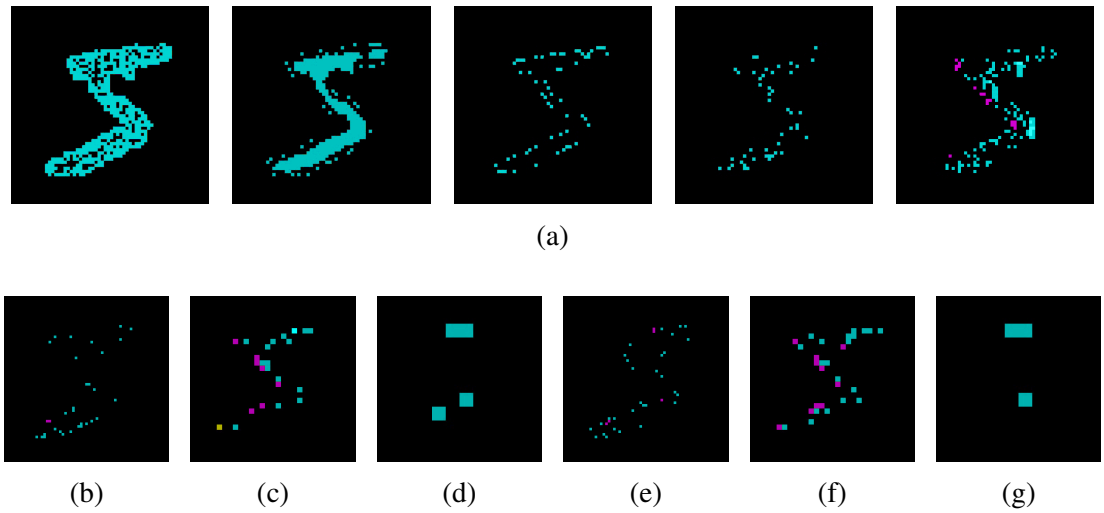
Figure 5.14: Formation of centre-surround receptive fields. a) Receptive field for bipolar (input) cells, kernel width = 7, $\sigma = 0.7$. b) Cross-correlation of input kernel is the connectivity weights for amacrine cell. c) By subtracting the cross-correlation kernel from the input one, we can form a centre-surround receptive field.

We first test the network without lateral competition (Figures 5.15b, c, and d), the centre-surround mechanism alone reduces spike counts to an average of 4.1% when compared to the total input spikes. Allowing different ganglion cell types to compete produces a further reduction in spike output (an average of 3.32%) and, most importantly, discourages the output representation from containing redundant information. Furthermore, with this competition we can transform the input from a rate encoding to a temporal one as the ganglion whose receptive field is a better match to the input

Table 5.1: Bipolar receptive field parameters.

Type	Width	σ	Stride (sampling step)
1	3	0.57	1
2	7	0.8655	3
3	15	1.3535	7

will fire first and inhibit neighbouring ganglions. This happens continuously and allows ganglion cells with a receptive field with the current best fit to spike, creating a sequence of "best-fit" spikes.



(a)

(b) (c) (d) (e) (f) (g)

Figure 5.15: Centre-surround response at different scales. In every image cyan and magenta pixels indicate activity in the ON and OFF path, respectively. Top row pictures show 20ms activity in the photoreceptors each, these are the input to the retinal model. Images in the lower row are obtained by accumulating spikes for 100ms after the filtering stage and at different scales: b and e, c and f, d and g; correspond to a 1:1, 1:9 and 1:49 scales, respectively. Figures b, c and d show the behaviour of the network without lateral competition; while e, f and g depict the behaviour with lateral inhibition.

Gaussian receptive fields in bipolar cells are just an approximation to the real weight distribution in the eye, these are most likely expressed in a distance-dependent connectivity rule in nature, for both bipolar and lateral competition connections. Evolutionary processes must have led this to maximize information while minimizing signals.

Table 5.2: LIF parameters for static visual processing

	V_{thresh}	V_{rest}	V_{reset}	C_m	τ_{mem}	τ_{ref}	τ_{syn}
Value	-55.0	-65.0	-70.0	0.25	10.0	5.0	2.0
Units	mV	mV	mV	nF	ms	ms	ms

Orientation detection

A subset of ganglion cells in the retina react predominantly to (moving) bars [Barlow et al., 1964; Zhang et al., 2012]; furthermore, some neurons throughout the V1 area of the visual cortex present a similar behaviour [Hubel and Wiesel, 1963]. We present a network architecture, which follows the same competition principle as the multi-scale one and whose objective is to detect bars rotated –oriented– at multiple angles. Different orientations are detected by classes of bipolar cells, their receptive fields are modelled as Gaussian ellipsoids which we obtain from the general formulation,

$$G_o(\vec{x}) = \frac{1}{(2\pi)^{n/2}\,|\mathbf{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}\vec{x}^{\top}\mathbf{\Sigma}^{-1}\vec{x}\right] , \tag{5.11}$$

where $|\cdot|$ is the determinant operator and the dimension of the system is given by $n = 2$; similarly, $\vec{x}$ and $\mathbf{\Sigma}$ are of dimensions 2×1 and 2×2, respectively. Since Σ^{-1} is symmetric and positive definite, a singular value decomposition (SVD) is guaranteed,

$$\mathbf{\Sigma}^{-1} = \mathbf{R}\,\mathbf{S}\,\mathbf{R}^{-1} . \tag{5.12}$$

We can interpret $\mathbf{R}$ as a 2D rotation matrix [Spruyt, 2017],

$$\mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} , \tag{5.13}$$

where θ is the rotation of the ellipsoid with respect to the positive horizontal axis. Matrix $\mathbf{S}$ can be seen as scaling factor [Spruyt, 2017],

$$\mathbf{S} = \begin{bmatrix} 1/s_x^2 & 0 \\ 0 & 1/s_y^2 \end{bmatrix} , \tag{5.14}$$

plus we can think of s_x and s_y as the width and height of the ellipsoid, respectively. We can now reformulate Eq. 5.11 into

$$G_o(\vec{x}, s_x, s_y, \theta) = \frac{1}{2\pi\left|\left(\mathbf{R}_\theta \mathbf{S}_{sx,sy} \mathbf{R}_\theta^{-1}\right)^{-1}\right|^{1/2}} \exp\left[-\frac{1}{1}\vec{x}^{\top}\mathbf{R}_\theta \mathbf{S}_{sx,sy}\mathbf{R}_\theta^{-1}\vec{x}\right] , \tag{5.15}$$

which allows us to produce arbitrary Gaussian ellipsoids. Receptive fields of orientation-detecting bipolars are defined using the parameters in Table 5.3 and the results are shown as the top row of Figure 5.16a.

Table 5.3: Orientation detection parameters.

Angle	Width	S_x	S_y	Sampling step
0, 45, 90, 135	7	3	0.5	3

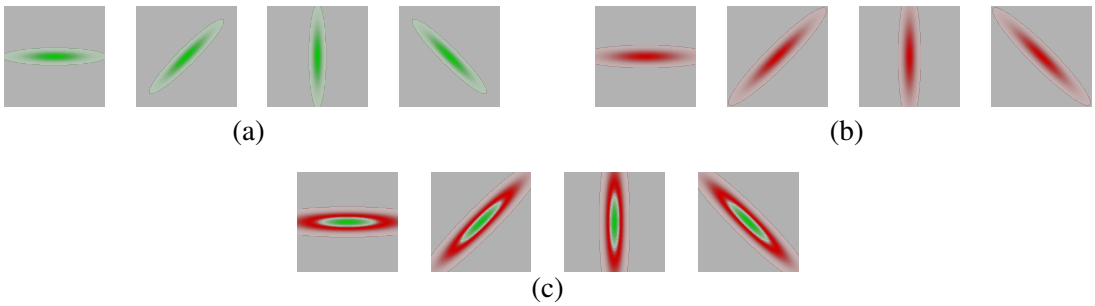
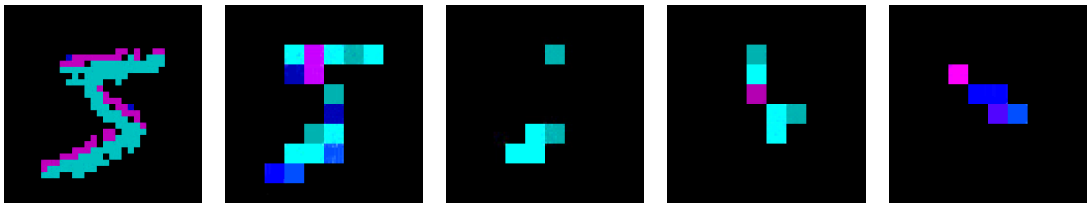


Figure 5.16: Orientation detection receptive fields. In all images green pixels are positive values and red ones are negative. Top shows the receptive field of bipolar neurons which are to detect oriented bars at four different angles (0, 45, 90 and 135). Auto-correlation of the bipolars' receptive fields are depicted in the middle row and the centre-surround-like receptive fields of ganglion cells constitute the lower row.

Similarly to the general image representation network, there could be redundant feature detection and reducing these is done through lateral inhibition. As in the previous network, inhibitory weights are computed by the auto-correlation of bipolar receptive fields (shown in Figure 5.16b). Since the input kernels are Gaussian, then the auto-correlations are also Gaussians but with a wider area; so the subtraction of input and inhibition gives rise to behaviour similar to centre-surround.

We tested orientation detection units (parameters can also be found in Table 5.2) by presenting a spiking representation of an MNIST digit (Figure 5.17a) to the network in which different orientation filtering neurons are simulated in parallel. We obtain higher responses from horizontal filtering neurons as more of the input matches this orientation; vertical filtering also obtains high responses though in localized regions, particularly where the orientation of the curves changes. Tilted orientations show better selectivity, this happens because the discretization of the receptive field renders high weights in one the diagonals of the kernels.



(a) (b) (c) (d) (e)

Figure 5.17: Orientation filtering results. In every image cyan and magenta pixels indicate activity in the ON and OFF path, respectively. The leftmost image is a sample of the input spikes, the rest are the accumulation of spikes throughout the trial. From left to right, the angle filtering is done at 0, 45, 90 and 135 degrees.

We understand the output of this "sub-network" as a different representation of the input, the resulting activity could be interpolated to obtain an estimate of intermediate orientation angles. For instance, if we see the same activity from the 45 and 90 degree filters, we may have an orientation of 67.5 degrees.

Motion sensing

Early motion sensing in animals allows them to react quickly to moving stimuli; furthermore, there are theories suggesting that motion cues could bring sufficient information to enable some invariance in V1 regions of the visual cortex [Földiák, 1991]. In mammalian retinas, *starburst* amacrine cells (SAC) react to moving objects and have a particular connectivity [Borst and Euler, 2011; Euler et al., 2014]. Firstly, instead of sampling activity directly from photoreceptors, they do so from bipolar cells but still output to ganglion cells. Another particular connectivity for these cells is that overlapping regions of dendritic trees from opposing direction-sensing cells inhibit each other. It is theorized that increasing delays in dendritic tree regions allow the cell to sum incoming voltage pulses if most arrive around the same time.

The design for motion detection units (MDUs) relies on the inherent properties of spiking neurons: line delays, input integration and leakage. Similarly to SACs, the premise is that if an object is moving, we can detect activation in a set of photoreceptors at one time and later detect another set; if we delay the output of the first set to coincide with the arrival of the second, we can assume there was a movement from one region to the other. Of course, this is a terrible assumption if only two points (pixels) are sampled as the sequence of detection could have originated from other sources (e.g. randomly flashing lights).

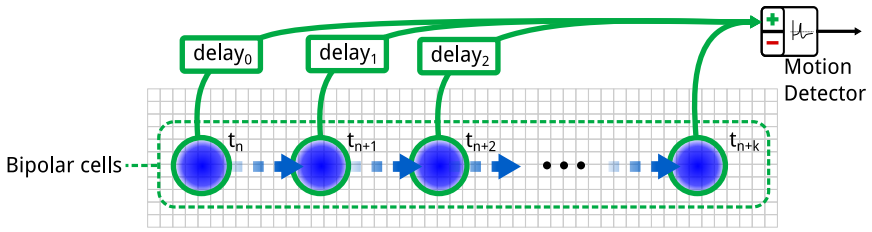


Figure 5.18: Multiple events occurred in sensed areas in a sequence.

If we add sampling points (Figure 5.18), it becomes less likely that all the spikes were originated by random events. Nevertheless, motion is detected in small windows so we can detect only apparent motion, thus movement of full objects would have to be detected in later stages of the pipeline when more information is assembled.

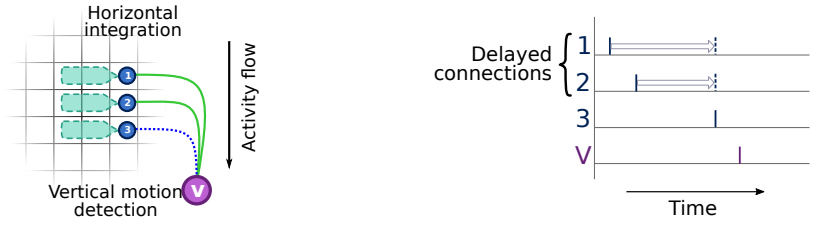
Table 5.4: Rounded average spike counts per direction sensing filters. Speed is in pixels per frame, at 90 frames per second.

	Direction			
Speed	East	North	West	South
1/4	11	0	0	0
1/2	13	0	0	1
1	32	1	1	1

To test our motion detectors we simulated movement of objects by translating an image (an MNIST digit) in the east direction at different speeds (see Table 5.4) and present it to the highest resolution filter in our general representation network (Section 5.2.1). We connect east, west, north and south MDUs to the bipolar cells of the filter and use centre-surround connectivity on the output of the detectors. At each trial of the experiment we count the spikes generated by each of the detectors; it can be seen in Table 5.4 that at each speed, the east MDU spikes the most. We could additionally tune more input connections to target different speeds so that the same neuron is sensitive to any motion perceived by the retina.

One problem with the previous model is that multiple activations of the same sampling point (e.g. a faulty, always-changing pixel in a camera) would still elicit a false motion detection event. To reduce false positives and target different speeds a closer approximation to the Reichardt detector is required, in which we have two neurotransmitters with different temporal dynamics [Borst and Euler, 2011]. One will be slow,

which can be seen as opening a window in which the post-synaptic neuron can spike. The second transmitter will have fast dynamics so it will be effective only for a short time. We added some more structure to the connectivity of the motion sensor, bipolar neurons sense either horizontally or vertically aligned regions of the input image (blue units in Figure 5.19a) and they have long refractory periods.



(a) (b)

Figure 5.19: Structured connectivity for motion detection.

Slow neurotransmitters are delivered by delay lines to maximise their effect on the post-synaptic neuron (green lines in Fig. 5.19b) by the time activity reaches the sampling zone for the bipolar neuron which will emit the fast neurotransmitter. In the motion sensing neuron the total of slow and fast inputs are multiplied, the product is the final input given to the neuron

$$I_{total} = I_{slow} * I_{fast} \; ; \tag{5.16}$$

where I_{slow} and I_{fast} are the current provided by the slow- and fast-changing neurotransmitters, respectively. Finally, the model for the motion-sensitive spiking neuron is based on the LIF model (Eq. 2.21) and is given by

$$\tau_m \frac{dV}{dt} = R_m I_{total} - V + E_L \; . \tag{5.17}$$

Figure 5.20 shows the results of giving the preferred and opposite sequence of events to the motion sensor. Membrane voltage is shown in magenta on the left plots. Plots on the right show the input to the neuron: blue and green are the fast and slow neurotransmitters, respectively; red is their multiplication which is the input current given to the neuron. It takes 500ms for the slow transmitter to decay while the fast one drops to zero in 10ms.

It can be observed that, for the incorrect sequence (Figure 5.20b), membrane voltage is a low-pass version of the total input, but never reaches the threshold since the

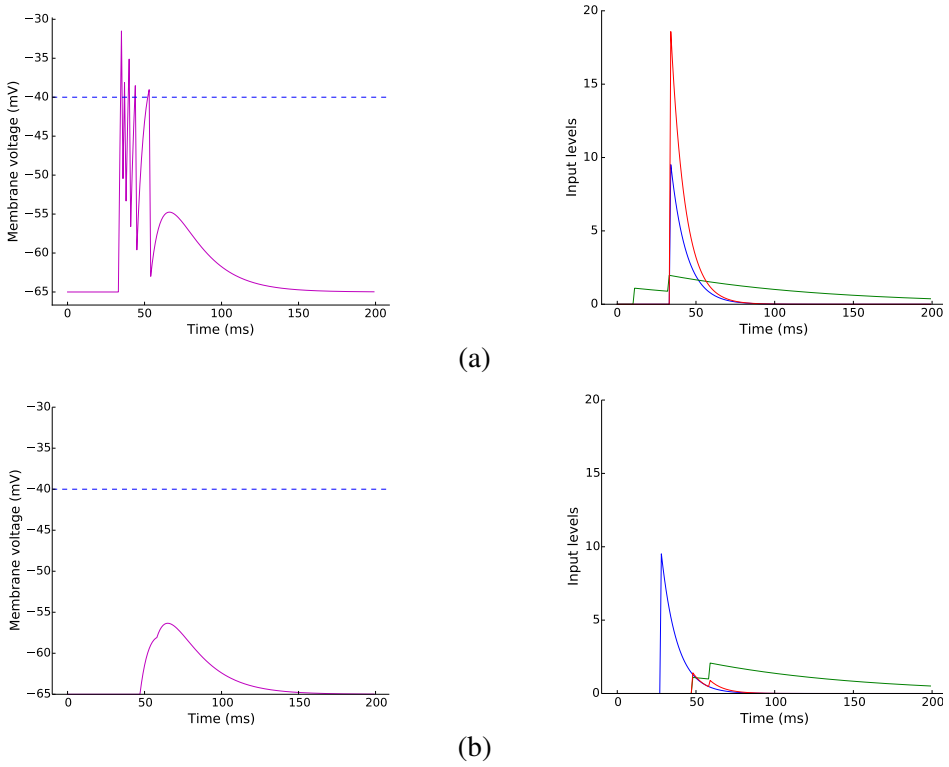


Figure 5.20: Motion sensing neuron dynamics.

fast input is nearly gone when the slow ones reach the neuron. In the case of the preferred sequence (Figure 5.20a), both transmitters reach their maximum at the same time and, since they get multiplied, there is a massive input to the neuron. This results in a burst of spikes for about 25ms; this is the result of the temporal dynamics of the neurotransmitters and the lack of a refractory period in the output neuron. If the motion detector gets configured with a non-zero refractory period, we can control the spiking behaviour, even to the point of having a single spike per detection.

This model was tested using a bouncing ball simulation (Figure 5.21), the ball was

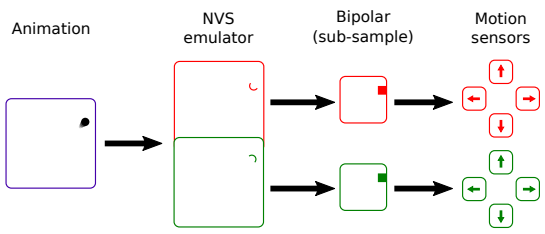


Figure 5.21: Network used to test motion sensing neurons

rendered onto a 5×5 pixel image and this was moved in a 64×64 environment. When the ball reaches a wall it bounces off with a random speed ranging from 1 to 2 pixels/frame in either direction. The sequence of raw images were processed by the NVS emulator and the resulting spikes were fed into a single-scale image representation circuit (Section 5.2.1). From this stage we obtain a lower resolution (half the width and height) version of the input, which helps the motion detector circuit to sense a single event per frame in its input. The temporal decay constants for the slow and fast inputs of motion-detecting neurons are $\tau_{slow} = 500ms$ and $\tau_{fast} = 2ms$, respectively; neural parameters can be found in Table 5.5.

Table 5.5: Neuron parameters for motion sensing.

	V_{thresh}	V_{rest}	V_{reset}	C_m	τ_{mem}	τ_{ref}	τ_{fast}	τ_{slow}
Sub-sample	-55.0	-65.0	-130.0	0.25	1.0	10.0	1.0	–
Motion	-55.0	-65.0	-100.0	0.25	10.0	10.0	2.0	500.0
Units	mV	mV	mV	nF	ms	ms	ms	ms

Note that for the sub-sample population both inhibitory and excitatory synapses have a temporal constant ($\tau_{syn} = \tau_{fast}$) of $1ms$ and they do not posses slow synapses. Figure 5.22 shows the outputs of easterly and westerly motion detection as red-dashed and green-solid lines, respectively. Ball position is indicated by blue dots in the plot: it moved towards the north-east for about 500ms, then it bounced off a corner and moved in a south-westerly direction until ~1250ms; finally, it takes off to the north-east again.

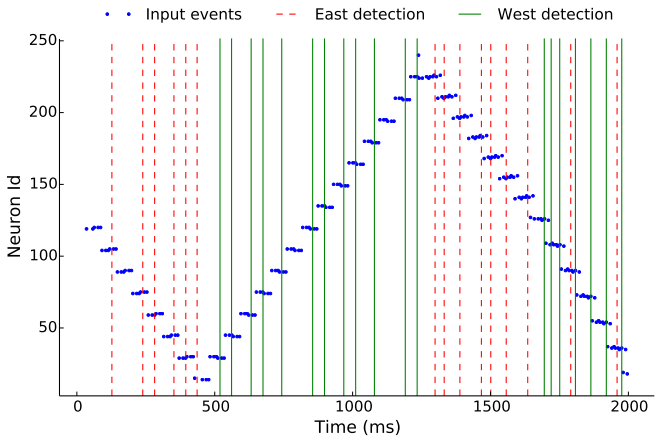


Figure 5.22: Output of motion sensing circuit in the horizontal axis of the ON channel.

In the first part (0 to ∼1250ms) of the experiment detection is near perfect although there are moments when detectors fail to sense motion. In the last section (after ∼1250ms) there are multiple false positive detections which can be diminished by lateral competition of different directions.

While both models detect motion, the first one will require multiple copies of the same circuit to detect different speeds. The latter one needs fewer copies as the decay time of the slow neurotransmitter can be used to group multiple speeds. In later stages of the pipeline, these groups can be evaluated to approximate the real speed of the visual stimulus.

Table 5.6: Motion detection effectiveness, true vs false positive percentage for bouncing ball and using 2-neurotransmitter detector

	Direction				Total
	East	North	West	South	
True Positive	73.94%	66.32%	63.59%	69.14%	68.24%
False Positive	26.06%	33.68%	36.41%	30.86%	31.76%

Mixing ON-OFF channels

Since the previous stage in the pipeline (the retina in biology) has separate channels for light change direction (increment or decrement), we may have the same information coming in both. We now take inspiration from the Lateral Geniculate Nucleus (LGN), which is thought to be a relay station and also a modulator. This stage of the pipeline corresponds to the relay aspect of the LGN, and aims to remove redundant information to reduce energy consumption while preserving the input. Neurons in LGN are distributed in layers which correspond to classes of retinal ganglion cells. The connectivity motif for these neurons is as follows: they receive excitatory input from one channel and inhibitory from the opposing one and from neurons in their own layer (Figure 5.23).

As in every component in the pipeline there is topographic correspondence of neurons from the retina and the LGN. Retinal ganglion cells ($Retina_{(x,y)}$) will only inhibit relay cells in the LGN which are in its neighbourhood ($LGN_{(x,y,\Delta)}$). This mechanism would permits the LGN to increase the perceived contrast ratio since the same location can not be active for both channels.

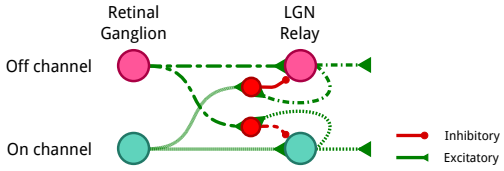


Figure 5.23: Cross-channel redundancy removal

During our tests there is little spike reduction, most likely due to the simple input image and good filtering from the multi-scale representation network.

Activity measurements

An important benefit of changing the raw pixel representation into a multi-scale model presented in this thesis is reducing activity, we define this as a measure of how many units are active out of the total at any given time step.

$$activity = \frac{active}{total} \times 100\% \tag{5.18}$$

Firstly, the event-based pixels in the NVS emulator are active only when the input image presents sufficient changes and this highly reduces activity. Furthermore, since we transformed the input into two channels (ON and OFF) the number of possible active units is duplicated. This transformation could reduce activity at least by two: when all pixels from the original image are active, the NVS would at most sense change in 50% of its sensors, but this comes at the cost of increasing pixel count. Measurements done with the MNIST digits show an average of 15% pixels in the input image are active (non-zero); after the NVS emulator an average of 2% neurons show activity per frame period (a 10ms window).

The filtering stages further reduce the activity and competition between scales push the active neurons to be the better-representing ones (i.e. large areas are represented by low resolutions while small regions by higher resolution kernels).

Table 5.7 shows how many active neurons, on average, are present through the pipeline; *Raw pixels* refer to non-zero-valued pixels seen in the original image. The *NVS emulator* column shows the average number of active neurons when transforming this technique to convert static images to spike trains; this is due to the nature of the transformation as only pixels with a large brightness change will be active. When using a multi-scale representation, we can further reduce the number of active neurons

Table 5.7: Average activity measurements through pipeline: MNIST digits.

	Raw pixels	NVS	Multi-scale	Cross-channel
Active	151	39	27	23
Total	1024	2048	2264	2264
Activity	14.74%	1.9%	1.19%	1.01%

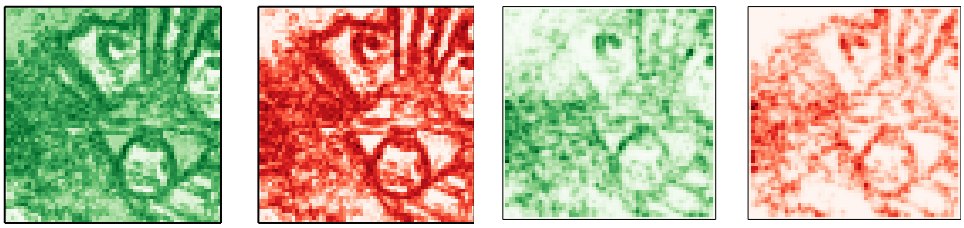
as each represents an image patch; checking for duplicated signals in ON/OFF channels did not reduce activity as much as previous stages.

One problem with traditional computer vision is that, even if only 15% of the pixels are non-zero, we need to perform operations on the whole image (e.g. convolutions). With event-driven computation this is not the case as only active pixels trigger operations, thus, reducing energy consumption. For example, the most common operation in CNNs are convolutions and we are required to perform multiple operations on vision datasets, such as MNIST. If the initial convolution kernel size is 7×7 pixels and the input image size is $28x28$, the number of multiplications required is about 38K. Taking the average active pixels for the NVS-representation this is reduced to around 2K multiplications.

Statistics shown in Table 5.7 were computed using the MNIST dataset and activity measurements are likely to increase for natural images, as they are more complex visual stimuli. Moreover, MNIST digits are easily represented with three scales of DoG filters; this number of scales is also unlikely to suffice in the case of natural images. We tested the same pipeline with images available in the SciPy package [Jones et al., 01] scaled to a 64×64 pixels resolution; results are presented in Table 5.8.

Table 5.8: Activity measurements through pipeline: SciPy images.

	Raw Pixels	NVS	Multi-scale	Cross-channel
Active	3694	828	676	134
Total	4096	8192	9768	9768
Activity	90.19%	10.1%	6.92%	1.38%



(a) (b)

Figure 5.24: Input vs output spikes. a) Spikes coming out of the NVS emulator. b) Reconstruction using the spikes coming out of the multi-scale network.

Since no feedback mechanism is present in our encoder we cannot adjust the excitability of neurons to better reflect the input then some information loss is bound to happen. This is shown in Figure 5.24, though we can appreciate that main characteristics are still present.

Summary

Biology has invested development in retinas and, as a result, they are complex neural structures. There must be an advantage that vision pipelines could make use of; particularly for neuromorphic hardware. We developed methodologies to transform conventional computer vision datasets into spiking representations. The main advantage is we can reduce the number of pixels which require further computation. In the case of the rank-ordered NVS emulator we see close to 10% active pixels with respect to the input image size, which roughly translates to 3.5% of the output pixel being active.

In our general image representation network we use Gaussian functions as input kernels, this resembles how bipolar cells sample photoreceptors. Additionally, feed-forward inhibition signals are sent from bipolar cells to neighbouring ganglion cells. In this way we can explain the usual model for retinal output, centre-surround receptive fields; furthermore, we also achieve temporal coding of input images, departing from the commonly used spike-rate coding. Moreover, by using Gaussian receptive fields, instead of centre-surround ones, we could reuse some of the bipolar cell output to combine with other types, such as motion sensing [Pack and Bensmaia, 2015; Roska and Meister, 2014].

By changing a standard image representation into a multi-scale, orientation and

motion one, we are increasing its dimensionality. Furthermore, enforcing competition between scales reduces the activity and the new representation is sparse (typically, for MNIST digits, 0.04 % are active per time step).

Although our current multi-scale network is purely feed-forward; we believe feedback signals could help produce a neural, compressed-sensing algorithm. Inter-plexiform cells in the retina receive input from amacrine cells and output to horizontal cells. Since amacrine cells receive input from bipolar cells (or other amacrine), this could be a biologically-plausible feedback mechanism to implement a full matching pursuit algorithm.

By slightly adjusting neuron and synapse models we were able to produce motion detection units which are believed to be key elements in biological vision. This shows how the temporal capabilities of spiking neurons provide great flexibility when modelling networks to process spatio-temporal inputs.

While transforming images to event-based representation helps improve energy efficiency by not having to process the whole image, in the SpiNNaker platform this is bound to how image-representing populations are mapped and connected. Current neural population partitioning and placement in SpiNNaker is done as if populations represent a 1D object. We should have different parameters for 2D or 3D, distance-dependent, topologically-mapped populations, so that traffic is constrained and post-neurons can process incoming spikes which correspond to distance limits instead of everything in a single line.

In the following chapter we explore how to learn the statistics of spiking visual representations.

Chapter 6

Biologically-plausible supervised learning

The cortex (neocortex) is the latest evolutionary addition to brain development. It is thought to be responsible for high-level behaviour such as language, long-term prediction and planning, to name a few. Mammals of different species assign cortical resources to their primary sensory or actuator organs (e.g. eyes/vision and hands for hominids, respectively) [Krubitzer and Seelke, 2012].

While the beginning of the visual pathway (up to V1) is thought to be static in adult mammals, some cortical connections are plastic and therefore give raise to learning. In extreme cases plasticity can reassign resources, to an extent, when sensory organs are missing. Furthermore, scientists have successfully made use of the plastic nature of the cortex to assign vision-like responses through tactile input [Kupers and Ptito, 2004].

The visual cortex creates conceptual abstractions of its input (see Chapter 5) by means of a hierarchical organization (Chapter 3.4). At the bottom of the hierarchy neurons respond to lines of different orientations; at the topmost level, populations of neurons react specifically to a concept (e.g. a face [Quiroga et al., 2005]). How this behaviour is achieved is still an open question, nonetheless, synaptic plasticity is surely involved.

In this chapter we develop the final stage of our spiking visual pipeline; to do so, we explore algorithms which modify synaptic weights in a supervised manner. Firstly, we demonstrate an implementation of an event-driven model of dopamine modulation on the conventional STDP learning rule. We then use this implementation to train SNNs with a dopamine-like signal to direct what a neuron should learn. Secondly, we take

a different approach to synaptic plasticity in SNNs which is still compatible with the STDP algorithm and is based on the work by Bengio et al. [2015]. Instead of using pre- and post-synaptic spike times we make use of the change in membrane voltage to estimate the how much weights should change. We take this last algorithm further and add an slow-decaying signal to perform supervision in SNNs.

Models of synaptic plasticity

The structure of cortical units is a first step towards solving the problem of a visual pipeline. For these units to generate the statistics of their inputs, we require a way to modify the connectivity of the networks. One method to achieve this is to change the strength, and influence, of synapses; this is usually known as learning. In the artificial intelligence literature, learning algorithms are typically classified depending on the required signals to alter synaptic weights:

Supervised learning

In this type of learning algorithm a separate entity -a supervisor- knows to which category the current sample from the input data belongs. In ANNs, a disparity between the category specified by the supervisor and the one produced by the network provides the error signal for algorithms such as back-propagation to drift the model towards the correct one. These algorithms are usually biologically implausible since, for example, neurons require non-local information to perform weight updates Whittington and Bogacz [2019].

A simple way to achieve supervised learning in SNNs is to initialize weights to such a small value that post-synaptic neurons will not spike, even if all of their inputs are active. When the supervisor acknowledges that the input belongs to a post neuron, it will send a signal which will increase the activity of the post-synaptic neuron. Forcing the post neuron to spike after the input will create a causal relation between pre- and post-synaptic activity. If the STDP algorithm (Section 6.1.1) is used, these spike pairs will tend to increase the weights for the corresponding synapses; thus, learn the statistics for a given input class. This is not ideal, as learning has to be turned off when the user deems it necessary else the network could diverge, and forget the “correct” weights. There are other methods for supervision in SNNs, they follow Hebbian principles and generally optimize spike timing. Some have to do a back-propagation-like

step to adjust synaptic efficacy, which is biologically implausible [Bohte et al., 2000] Researchers have also added a special input type which can tell the post-synaptic neuron when it should fire. If the latter spikes at an undesired time, it will adjust the weights for incoming connections [Ponulak and Kasiński, 2010]

Unsupervised learning

Training algorithms in which the system will acquire statistics from the inputs without labelled data constitute the category of unsupervised learning. ANNs best known algorithms are the Restricted Boltzmann Machine (RBMs) [Hinton et al., 2006; Tieleman, 2008], Auto-encoders (AE) [Bengio et al., 2009; Goodfellow et al., 2016] and Generative Adversarial Networks (GANs) [Goodfellow et al., 2016].

RBMs are undirected, two-layered, binary unit graphs where changes to the connection weights are achieved by pushing them towards the input distribution and away from the current state of the network. The input distribution is visible since we have samples, but the underlying distribution is not and computing it is intractable, so Monte Carlo (MC) methods are required to estimate it. Contrastive Divergence (CD) is an efficient algorithm to train RBMs in which the estimate for the underlying distribution is obtained after n steps of Gibbs sampling.

The structure of AEs is similar to RBMs but the graph is directed and we have a visible (input), a hidden and a reconstruction population. The main idea is to generate a sample from the hidden population given an input; then the sample from the hidden population will be used to reconstruct the input. An error between the input and reconstruction is calculated and this is used in back-propagation to compute the required weight change. Although the input-to-hidden (feed-forward) weights could be independent from the hidden-to-reconstruction (feed-back) ones, in practice making the feed-forward weights the transpose of the feed-back works fine and accelerates training. There is a risk of just learning the identity function so strategies such as de-noising are used to generate more robust features than with the standard algorithm.

Training for GANs is based on game theory. Basically a generator attempts to create an image and a discriminator judges whether it is real or fake. The generator samples from a simple distribution for the hidden state and outputs an image through a neural network. The discriminator, another neural network, receives the generated image and a sample from the input distribution; the error is calculated by comparing the inputs and attempting to be correct in judging whether the generated input is close enough to the real one. Later, the error from the discriminator is used to transform the

generator network pushing it towards one which will create fake inputs close to the real ones.

In SNNs the most utilized algorithm, STDP, is already unsupervised; it attempts to build a model for input statistics by finding correlations in spike times. The algorithm is based on Hebbian principles: the connection weight is increased if there is a causal relationship between pre- and post-synaptic activity, that is, the pre spike occurred before the post one; to reduce non-causal spike pairs, the connection efficacy is reduced if the post spiked before it received the pre spike. There are many variants to the rule which go from coincidence detection, to causality-only, to reduced precision.

Reinforcement learning

Supervised and unsupervised learning are good methods to separate inputs into classes but are not ideal to enable an agent to achieve a goal given input from an environment with which it interacts. The main difference in Reinforcement Learning (RL) is that training is done by adjusting the model from rewards and/or punishments provided by the environment given actions made by the agent [Sutton and Barto, 1998]. Furthermore, this learning is commonly done on-line and could adapt to new goals, or bind together a series of actions instead of a single category of objects.

Classical RL models are usually Markov decision processes, for which probabilities of transition are adjusted when rewards are granted. Recently there has been renewed attention to this approach as Convolutional Neural Networks (CNNs) have been combined with Q-learning to play video-games [Mnih et al., 2015]. Deep Q-learning (DQN) has achieved super-human level playing these games. A key aspect of the algorithm is that it keeps a buffer of previous transitions (replay memory) from which to chose at random. Another key element is to keep two CNNs one which is updated every iteration (online) and a target network whose weights are copied from the online network every K steps. The error function for the algorithm comes from the received reward and the difference between the target and the online network evaluations.

Since SNN learning rules usually depend on instantaneous pre- and post-synaptic neuron activity, implementing reinforcement learning requires a trace to keep these states on a much greater time scale (i.e. seconds instead of milliseconds). Additionally, a reinforcer input is required and this is typically applied as a modulator to STDP.

Spike-timing-dependent plasticity

In 1949, Hebb produced his insightful postulate:

> *When an axon of cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.*

This implies that the algorithm is local, as weight updates are a function only of the states of pre- and post-synaptic neurons:

$$\Delta w_{pre,post} \propto f\left(s(pre), s(post)\right) \tag{6.1}$$

where *pre* and *post* refer to the position of the neuron with respect to the synapse which connects them, w is the synaptic efficacy, and $s(\cdot)$ indicates the state of the desired neuron.

Hebbian learning can be interpreted as increasing synaptic efficacy, or long-term potentiation (LTP), for neurons whose activities are correlated [Gerstner, 2016]. Conversely, uncorrelated neurons' weights are usually decreased, also known as long-term depression (LTD), in most learning algorithms. In particular, spike-timing-dependent plasticity (STDP) follows Hebbian principles, the states can be abstracted just to the times at which pre and post neurons spiked [Bi and Poo, 1998; Markram et al., 1997].

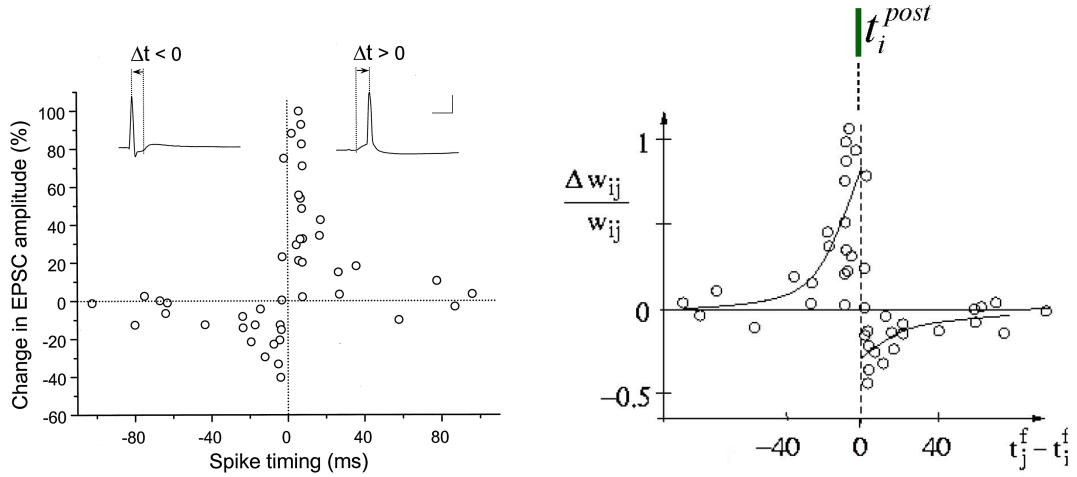
$$\Delta w_{pre,post}(t) \propto f(t, t_{pre}, t_{post}) \tag{6.2}$$

If the pre neuron spikes before the post one, the synaptic weight is increased; conversely when the post neuron spikes before it receives a spike from the pre neuron, the weight is decreased. This roughly establishes causality (or at least correlation) of the activities of the neurons.

Typically, temporal window curves (see Figure 6.1b) are modelled as exponential functions which decay as the temporal difference between pre and post spikes increase:

$$W_T(\Delta t) = \begin{cases} A_+ e^{-|\Delta t|/\tau_+} & \text{if } \Delta t < 0\,; \\ A_- e^{-|\Delta t|/\tau_-} & \text{if } \Delta t > 0\,; \\ 0 & \text{otherwise}\,. \end{cases} \tag{6.3}$$

In all cases Δt is the temporal difference of the pre- minus the post-synaptic spike



(a) (b)

Figure 6.1: a) Original STDP data. From Bi and Poo [1998]. b) Redrawn STDP experiment, normalized weight change and superimposed curves. From Sjöström and Gerstner [2010].

times. When Δt is negative it implies causality of the post spike, thus the weight should be increased and so $A_+ > 0$. In the other case, $\Delta t > 0$, will produce negative weight changes ($A_- < 0$). These effects are summed for every pre-post spike pair:

$$\Delta w_{pre,post} = \sum_{j}^{N_{pre}} \sum_{i}^{N_{post}} W_T(t_j - t_i) \tag{6.4}$$

Network architecture

As discussed in Chapter 4, cortical regions react to specific areas of their sensory input. Hubel et al. observed retinotopic responses and described "cortical columns" whose maximal activity corresponds to the presentation of specifically oriented, moving bars. It is unlikely that a hard separation of columns is present in the cortex but this abstraction helps the understanding of how neurons may be organized in cortical regions. As we move up in the visual cortex, topographic organization is maintained although abstraction and receptive field sizes are increased. At the top of the visual hierarchy neurons can obtain most information required to categorize the input. For the plastic stage of our visual pipeline we follow cortical organization principles:

- *Distance-dependent connectivity*. Neurons have limited (local) receptive fields

with respect to their immediate input: neurons in area A_n will have higher probability of connecting to nearby neurons in area A_{n-1}.

- *Hierarchical*. Determining a category for the input image requires neurons whose (global) receptive field can cover the full input. This makes a hierarchy of neurons –with increasing receptive fields– a necessity; furthermore, model composability properties allow hierarchies to use fewer neurons to represent the input statistics.

- *Simple-complex*. Simple neurons receive inputs mostly from a lower hierarchical level, usually containing distinct origins (e.g. ON/OFF channels, orientation angles). Complex cells sample from simple neurons, this allows them to join otherwise separate inputs.

- *Competition*. In most cases neurons should compete to represent the input through inhibitory signals; competition circuits can be set in a random, distance-dependent or hard-wired (e.g. winner-takes-all motif) manner.

Through most of the visual hierarchy, synapses are formed (and removed) and their efficacies are adjusted to achieve classification of input patterns. The most commonly used plasticity algorithm in SNNs is an unsupervised one (STDP) and it relies on stochastic behaviour of the system for neurons to specialize (or favour) an input pattern over another. Another key component for neuron specialization is local competition circuits which make weight changes steer in different directions for competing neurons.

Winner-takes-all

A common motif in SNNs is the Winner-Takes-All (WTA) circuit; here an active "winning" neuron (A in Figure 6.2) is chosen because it has a higher rate or fires earlier than the losing neurons (B and C in Figure 6.2). This neuron will inhibit others, that is it will prevent the other neurons in the circuit from producing spikes. Inhibitory signals are generated by inter-neurons whose response and refractory times are shorter than regular neurons. In some instances there could be more than one winner; this is usually referred to as a soft winner-takes-all (SWTA) circuit.

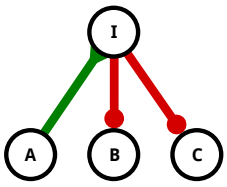


Figure 6.2: Example of activity for a WTA network, when neuron A is the winner and it inhibits neurons B and C.

Modulated learning

Since categorization of inputs is the main motivation for plasticity in our pipeline we depart from unsupervised algorithms (STDP) and investigate biologically-plausible methods for supervision. In the following sections we study different plasticity rules which guide neurons to represent specified concepts. In biology, control of plastic changes is associated with different receptor types; we reflect this in our SNN models by introducing distinct inputs (similar to the presented in Section 5.2.3).

Third-factor rules

Traditionally, simple models of SNNs have used two types of synapse: excitatory (positive - AMPA[1]) and inhibitory (negative - $GABA_{A,B}$[2]). These drive changes to the membrane voltage and, indirectly, can produce weight changes [Gerstner et al., 2014].

In biological neural networks, there are more neurotransmitters which may alter learning processes. Research on dopamine (DA) interaction shows that it could be crucial for reinforcement learning as it has been identified as a control signal for large regions of the brain. Similarly, NMDA[3] is a neurotransmitter which may be essential to supervised learning as it modifies neural and plastic responses [Clopath et al., 2010; Frémaux and Gerstner, 2016; Roelfsema and Holtmaat, 2018].

Furthermore, there are additional cells involved in synapse function; *astrocytes* are usually characterized as "maintainers" in the central nervous system as they keep ionic concentration stable, form scar tissue on damaged regions, and aid energy transfer [Sofroniew and Vinters, 2010]. Scientists are focusing more effort to understand the role of astrocytes as learning modulators [Gibbs et al., 2008]. Research shows that

[1] α-amino-3-hydroxy-5-methyl-4-isoxazolepropionic acid
[2] Gamma-aminobutyric acid
[3] N-methyl-D-aspartate acid

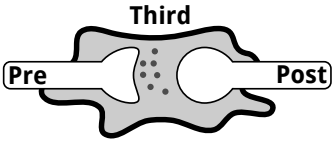


Figure 6.3: Cartoon of third factor interaction on plasticity.

these cells are also involved in the regulation of current frequency, short- and long-term plasticity, and synapse formation or removal [Halassa et al., 2007; Perea et al., 2009; Tewari and Majumdar, 2012; Theodosis et al., 2008].

Having a third component modifies Hebbian-based (equation 6.1) weight updates and now a rule will depend on the state of three parties (Figure 6.3):

$$\Delta w_{pre,post}(t) \quad \propto \quad h\left(s(pre), s(post), s(third)\right) \tag{6.5}$$

where $s(\cdot)$ indicates the activity or state of a party. If only the time is considered as the state –as is the case for STDP–,

$$\Delta w_{pre,post}(t) \quad \propto \quad h\left(t, t_{pre}, t_{post}, t_{third}\right) \tag{6.6}$$

where t_x is the time at which a spike from neuron x was perceived by the post neuron.

Ponulak and Kasiński introduced an STDP-like rule with a third factor (ReSuMe), the extra input is used to get the post neuron to spike at a particular time [Ponulak and Kasiński, 2010]. When $t_{third} - t_{pre} > 0$, a weight increment (Δ_+) is applied to synapses; whereas weight depression (Δ_-) is applied when $t_{post} - t_{pre} > 0$. If the post neuron activates at the desired time $(t_{third} = t_{post})$, depression will be equal to potentiation and the total weight change will be zero, this is,

$$\Delta_T = \Delta_+ - \Delta_- = \Delta_+ - \Delta_+ = 0 \ . \tag{6.7}$$

Nichols et al. developed a three-factor learning rule whose purpose is, also, to learn spike times; to do this the third input to the synapse carries a "temporal target" signal and will alter the magnitude and direction of the weight change [Gardner and Grüning, 2016; Nichols et al., 2017]. The main difference from the ReSuMe rule is that this requires, additionally, a low-pass filtered version of the error. The temporal error is to modify the effect a single pre-synaptic spike has on post-synaptic neuron activity. The filtered version of the error can be seen as an "accumulation" activity for a time

window ($\approx 10ms$).

While the previously mentioned rules make use of a third factor, they remain biologically implausible as a synapse is unlikely to be able to keep track of exact times. In this context, we can see the neurotransmitter dopamine as a global error signal, or a modulator which enables learning after the previous activity in the network led to a reward-worthy action [Sutton and Barto, 1998].

Other modulators (e.g. NMDA, serotonin, noradrenaline) could guide plasticity through attention-like mechanisms. These are thought to be local signals -as opposed to dopamine- and may represent feedback and/or lateral interaction [Roelfsema and Ooyen, 2005].

Models of modulated synaptic plasticity have been developed and in general follow

$$\Delta w_{pre,post}(t) \propto g(t, t_{third}) \times f(t, t_{pre}, t_{post}) \tag{6.8}$$

where f is the regular plasticity function (e.g. STDP) and g is the, usually, decaying response of the modulatory input.

Eligibility traces / synapse tagging

For reinforcement learning, a history of the plasticity function is required, usually called an *eligibility trace*. The intuition behind this mechanism is that events in the world occur at a lower speed than spike interactions and behaviour can be rewarded (or punished) only after it has happened. Furthermore, researchers have found some evidence of eligibility traces in biology [Fisher et al., 2017; Gerstner et al., 2018].

In mathematical models, eligibility traces "store" weight updates for a long period, until a signal arrives at the synapse which triggers "application" of the current state of the trace [Florian, 2007; Izhikevich, 2007b]. The signal could be dopamine, or another neuromodulator, which has slower dynamics, decaying in the scale of hundreds of milliseconds.

In Figure 6.4a the trace labelled *STDP* shows the weight change functions given the inputs shown in rows *pre* and *post*. Eligibility traces are formed by $< pre,\ post >$ spike pairs which are illustrated in zones 1 and 2 in Figure 6.4a; these cause accumulation of weight changes driven by STDP curves. Since STDP interactions depend on the time at which the pre and post neurons spiked, if these times are sufficiently far in time, no weight change is added to the eligibility trace (zone 3 with respect to zone 2 in Figure 6.4a).

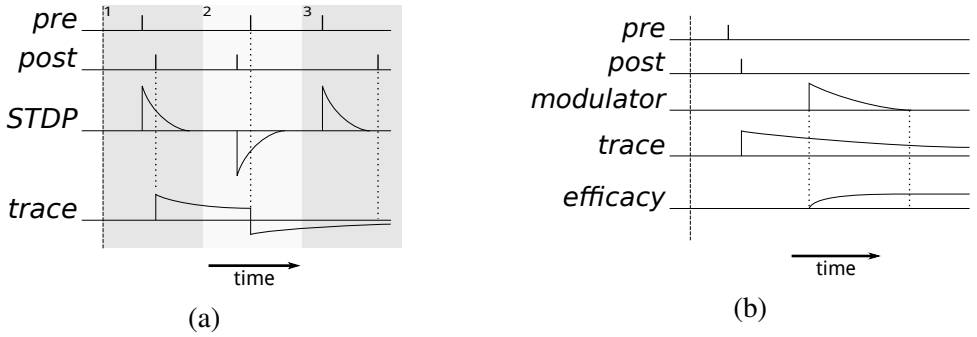


Figure 6.4: Eligibility trace.

Eligibility traces have much slower dynamics than STDP interactions as illustrated in Figure 6.4a; the curve in row *trace* decays much slower than any of the curves in row *STDP*. The decay rate is low which is useful to keep track of how temporally distant weight changes contributed to a particular behaviour.

The modulating neurotransmitter (*modulator* curve in Figure 6.4b.) also has slower dynamics than STDP, but not as slow as eligibility traces. Weight changes are only applied when the third signal is present, this is modelled as a multiplicative effect

$$\Delta weight(t) \propto modulator(t) \times trace(t) \text{ , or,} \tag{6.9}$$

$$\frac{dw(t)}{dt} = m(t) \times c(t) \text{ .} \tag{6.10}$$

The dopamine-based, modulated plasticity model as proposed by Izhikevich was implemented for the SpiNNaker machine [Mikaitis et al., 2018]. Eligibility trace dynamics are described by the following equation,

$$\frac{dc(t)}{dt} = -\frac{c(t)}{\tau_c} + STDP(\tau_{-/+})\delta(t - t_{pre/post}) \text{ ;} \tag{6.11}$$

where $e(t)$ is the state of the eligibility trace; $STDP(\tau_{-/+})$ is the value from STDP (Fig. 6.4a) curves and $\delta(t - t_{pre/post})$ is the Dirac delta function. Similarly, the modulator is governed by

$$\frac{dm(t)}{dt} = -\frac{m(t)}{\tau_m} + M(t)\delta(t - t_{mod}) \tag{6.12}$$

where $m(t)$ is the current state of the "local" modulating transmitter and $M(t)$ is the concentration of the "external" modulatory signal. In both cases, incoming signals create an instantaneous change (spike) thus the use of Dirac delta functions.

Since SpiNNaker is an event-driven computation platform, these equations required modifications. Weight changes are performed when a spike arrives at a post-synaptic core, thus the implemented weight update rule is[4]:

$$\Delta w(t) = \frac{1}{-\frac{1}{\tau_c} - \frac{1}{\tau_m}} c(t_{lc}) m(t_{lm}) \left[e^{-\left(\frac{te - t_{lc}}{\tau_c}\right) - \left(\frac{te - t_{lm}}{\tau_m}\right)} \right]_{t_{lw}}^{t}, \tag{6.13}$$

where le, lm and lw subscripts indicate the time (event) at which the last eligibility trace, modulator and weight updates where performed, respectively. As two different spike "types" can be received, weight updates will be performed either at $t_{lw} = t_{lc}$ or $t_{lw} = t_{lm}$. The terms inside square brackets come from performing a definite integral so we must evaluate the expression with $t_e = t$ and subtract it from the evaluation with $t_e = t_{lw}$.

Use cases

Modulation of plasticity rules can be used in different scenarios; the next sections describe tests done using the dopamine-based plasticity rule. Firstly, we evaluate whether the prompt release of dopamine into a population can increase the response to a particular input pattern. Secondly, we test whether dopamine signals could aid in the formation of Gabor-like receptive fields in a visually-driven network. Finally, a full visual pipeline is developed to classify MNIST digits.

Credit assignment

We tested the learning rule replicating an experiment which was originally published in Izhikevich [2007b], the author named it 'credit assignment'. The name comes from reinforcement learning terminology and this is because the reward signal should affect temporally close activity more than distant one. In the experiment, different groups of neurons in a population are stimulated at random times and one of the groups is chosen to be rewarded. The expected output of the experiment is to have larger weights on synapses originating from the chosen group and, as a consequence, higher activity when the group is stimulated.

The neural network for this procedure consists of 1000 neurons, divided into two

[4] See Appendix C for a derivation of the rule.

Table 6.1: IFCurrExp neuron model parameters for credit assignment experiment.

	C_m	I_{offset}	τ_m	τ_{refrac}	τ_{syn_E}	τ_{syn_I}	V_{reset}	V_{rest}	V_{thresh}
Value (Exc)	0.3	0.0	10	4	1	1	-70	-65	-55.4
Value (Inh)	0.3	0.005	10	2	1	1	-70	-65	-56.4
Units	nF	nA	ms	ms	ms	ms	mV	mV	mV

populations, they will emit either excitatory or inhibitory signals (*Exc* and *Inh* in Figure 6.5, respectively). Each neuron connects to others at random with a 10% probability, regardless of the neuron population "type" (red and green lines Figure 6.5).

Within the excitatory population we create groups with 50 neurons (5% of the total) chosen at random; the first group (S_1) is chosen as the pattern to search. We stimulate each group at random, maintaining a maximum of 5 groups spiking per second. Additionally, we inject "background" Poisson noise at $10Hz$, this puts neurons in a biologically-plausible setting. In terms of the credit assignment problem, we try to learn group S_1's activity pattern in a noisy environment, generated by other groups and random activity incited by background noise.

To learn the particular activity, we connect a modulator input (dopamine-like) to the excitatory-to-excitatory connections. Whenever group S_1 is stimulated, we send a dopamine pulse with a random delay in the range $[0,1)$ seconds; this will act as the reward signal.

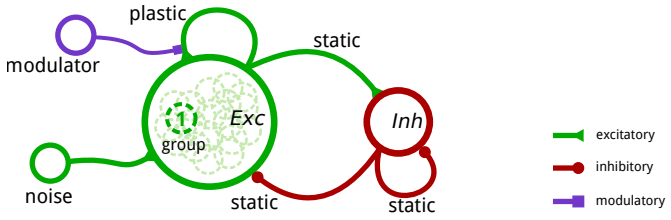


Figure 6.5: Credit assignment experiment network.

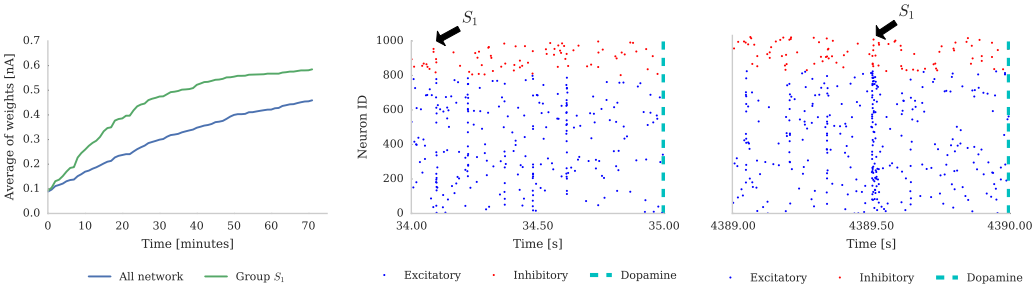
For this experiment we use standard LIF neurons, parameters are set to promote higher excitability from the neurons in the inhibitory population when compared to the excitatory one. High excitability is achieved by adding a small base current, reducing the distance between the resting voltage and the threshold value, and reducing the refractory period (see Table 6.1). We used exponentially-decaying, current-based synapses whose temporal constants (τ_{syn_X}) are set to 1ms to further approximate the original experiment.

Table 6.2: STDP parameters for credit assignment experiment.

	A_+	A_-	τ_+	τ_-	τ_c	τ_d
Value	1	1	10	12	1000	200
Units	–	–	ms	ms	ms	ms

The learning algorithm was parametrized so that the area for long-term depression (LTD) is 20% greater than the area for long-term potentiation (LTP) (A_+, A_-, τ_+, τ_- in Table 6.2). Dopamine interaction was characterized in a biologically-plausible manner: the temporal constant for the eligibility trace, τ_c, is 1000ms which implies the network will learn events that occurred up to a second ago; the temporal constant for dopamine, τ_d, is 200ms according to biological evidence.

We ran the experiment for about 1.5 hours of biological real time. At approximately 70 minutes the weights from group S_1 to other excitatory neurons are sufficiently large to be visibly noticeable in a raster plot.



(a) (b) (c)

Figure 6.6: Behaviour of the network in the credit assignment experiment. a) Average weight in the network, the blue line represents the average for all weights and the green one shows the mean weight for connections leaving neurons in group S_1. b) Spiking activity at the beginning of the experiment. c) Networks activity at the end of the experiment.

The evolution of the average weight in group S_1 is shown in Figure 6.6a as a green line, which presents a fast growth pattern. We can also observe the average weight value for every connection in the network as the experiment progresses (blue line), it grows but more slowly. Spiking behaviour for all groups is similar in the beginning of the experiment, this can be seen as correlated vertical dots in Figure 6.6b.

By the end of the experiment, connections which originate from group S_1 are at such value that most post-synaptic neurons will spike soon after group S_1 is stimulated. This can be seen in the middle of Figure 6.6c as a burst of activity. Although the network still responds to other patterns, it is now tuned to emit a higher response to the S_1 pattern. This has been observed in cortical regions, for example, a column in V1 will show a response to many oriented bars as inputs, but it presents the maximum spike rate for a particular orientation.

Learning digit patch prototypes

In this section we focus on learning regions of digits (patch prototypes) using plasticity control signals. In biology, dopamine is a global (i.e. reaching all neurons in a region) signal which can be thought of as enabling plasticity; here we remove the global constraint and target specific groups of neurons. Furthermore, our reward signal is not linked to the output of the network but to the input so that only the neurons which are assigned a particular input class are allowed to learn it. To test whether a plastic network could learn such visual patterns, we created a network whose input is the multi-scale representation (Section 5.2.1) for MNIST digits. The output of the multi-scale network is further filtered with the channel-mixing strategy presented in the same Section 5.2. The resulting activity serves as the input for two populations of neurons, one per channel (ON and OFF), in which N neurons in a region will be assigned to a specific digit. This is similar to the arrangement seen in the V1 area of the mammalian cortex in which particular "simple" cells receive input from just one channel.

Since we are trying to investigate learning regions of different digits, each region consists of $M \times N$ neurons; where M is the number of unique digits in the dataset (0, 1, ... 9) and N indicates how many neurons represent each digit class. Excitatory neurons in a region share a common input area, as shown in Figure 6.7a, though connectivity is initialized at random with a 50% probability. Additionally, neurons in the same region have been arranged in a Soft Winner-Takes-All (SWTA) circuit to promote diversity of learned weights (Figure 6.7b).

During the learning procedure we show each digit for 500*ms* followed by another 500*ms* period in which no input is given to the network. We apply a reward spike to the connections reaching the corresponding N neurons to the digit which is currently shown and this, eventually, specializes neurons. The time to give the reinforcement signal is chosen at random in the range $[50, 90)\%$ of the exposure period, this is so that

neurons have sufficient time to have spiked. Reward and STDP parameters for these experiments are presented in Table 6.3.

Table 6.3: Learning parameters for receptive field formation experiment

	STDP				Reward	
	A_+	A_-	τ_+	τ_-	τ_c	τ_d
Value	0.1	0.12	20	20	500	100
Units	–	–	ms	ms	ms	ms

We modified the time constant for the eligibility trace (τ_c) so that it covers at most the full exposure of a digit (500ms); we also diminish the time constant for reward synapses, τ_d, to 100ms. Finally, each pixel in the input is given a 5Hz Poisson noise source to further induce stochastic behaviour of the learning procedure. Even after a single epoch of the MNIST training set synapses are more efficient in the regions which show higher activity for their assigned digit. The sum of input weights (removing those which do not differ from the initial setting) for each digit class are shown in Figure 6.8.

As the learning procedure advanced we observed that weights for the middle scale representation settle first (Figure 6.8b), we speculate that this could provide stability for the other scales. The lowest resolution version of the input does not seem to provide much consistent activity, thus weights barely resemble digits (Figure 6.8c). However, most of the regions whose weights have high values do tend to have blob-like forms. Since noise was added to the input, the highest resolution weights changed often during initial training but, eventually, settle to border-like features (Figure 6.8a).

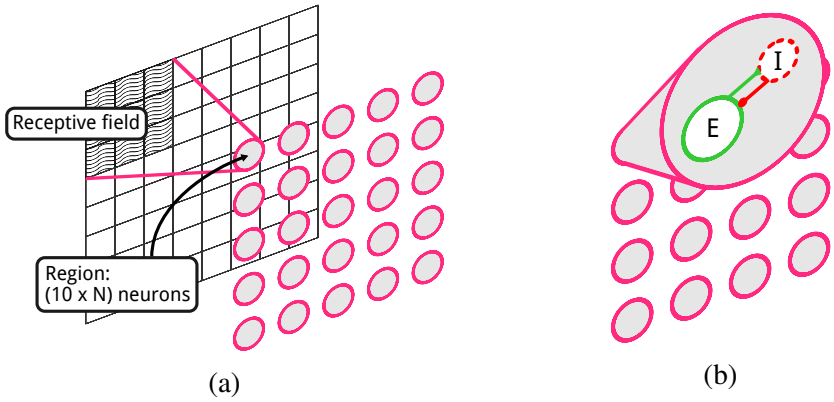


Figure 6.7: a) Distance-dependent network to learn regions of MNIST digits.

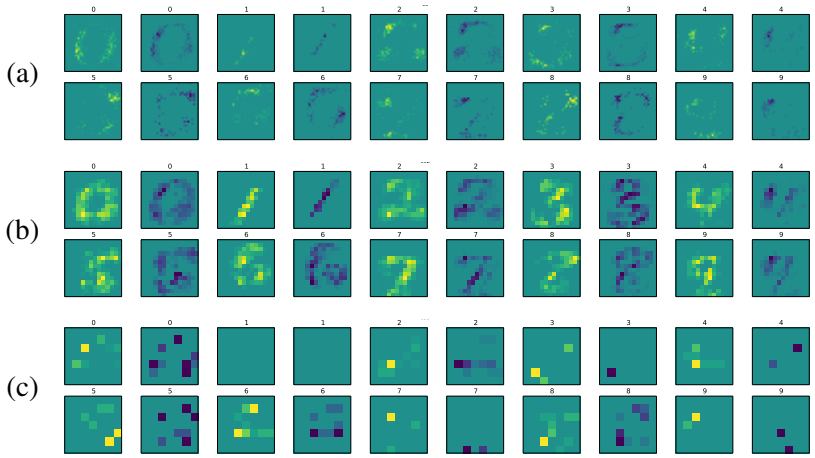


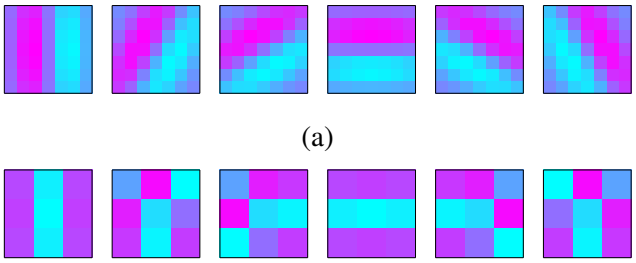
Figure 6.8: Sum of weights for neurons with the same class per channel. Weights perceived at the highest-, middle- and lowest-resolution correspond to Figures a, b and c, respectively. Left column for each class shows weights for the ON-channel-focused population and the right one to the OFF-channel one.

Figure 6.9: Hand-picked weight averages: dark indicates OFF and light corresponds to ON. The shape resembles some weights learned by other ANN procedures.

Finally, weights in different regions of the input have receptive fields whose weights resemble portions of digits (Figure 6.9) for the high- (left) and middle-resolution (right) scales. In some cases, particularly neurons whose receptive field are at the central region of the image, neurons learn complex patterns. In fact, in that central region, the neurons' reaction is higher for their corresponding class with a 27% accuracy (when compared to neurons assigned for other class).

Digit recognition

Since digit patch prototypes study resulted in low recognition accuracy we designed a more general network. The input for this network is a multi-scale representation of digits (as in the previous case) though using two scales only. Following the channel-mixing stage we added a set of convolution-like layers with Gabor kernels, this to have a general filtering stage (as opposed to digit patch prototypes). The latter resembles



(a)

(b)

Figure 6.10: Gabor filters for V1-like filtering. a) A 7×7 kernel is used for the high-resolution scale. b) A 5×5 kernel is used for the middle-resolution scale.

the V1 area of the visual cortex which is thought to be fixed even before eye-opening in mammals. Gabor filters were generated using the following equations:

$$O(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right), \tag{6.14}$$

$$x' = x\cos\theta + y\sin\theta, \tag{6.15}$$

$$y' = -x\sin\theta + y\cos\theta. \tag{6.16}$$

Where λ and ψ are the wavelength and phase of the sinusoidal component, respectively; θ is the orientation of the resulting stripes, σ is the standard deviation of the Gaussian component; and γ the spatial aspect ratio. Parameters for the generation of Gabor kernels are presented in Table 6.4.

Table 6.4: Gabor filter parameters

Resolution	Width	Sampling	σ	λ	γ	ψ	θ
High	7	2	2	6	0.5	1.7	[0, 30, 60, 120, 150, 180]
Middle	5	1	2	6	0.5	1.1	[0, 30, 60, 120, 150, 180]

Following V1 connectivity patterns the negative values from Gabor kernels (magenta pixels in Figure 6.10a) have been rectified and are used as synaptic efficacies to receive input from OFF-channel neurons. Similarly, positive values were used in the connections from ON-channel neurons to V1 populations.

To diminish redundancy of representations of the same feature we make use of local lateral inhibition (an analogue of centre-surround behaviour of the multi-scale

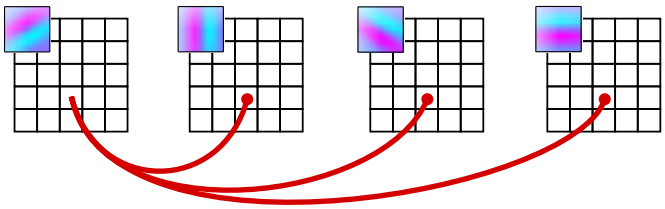


Figure 6.11: Competition among different feature detectors.

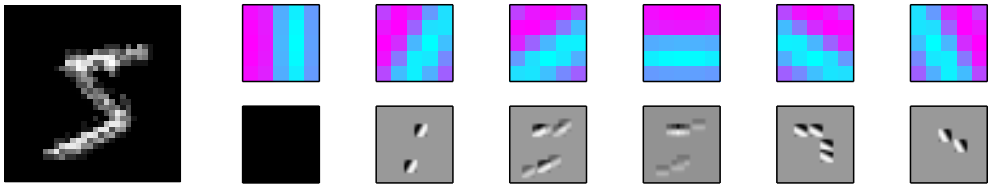


Figure 6.12: Results of feature detection stage

representation) through a Gaussian inhibitory kernel. Additionally, competition between features is also promoted using Soft Winner-Takes-All (SWTA) circuits in a per-neuron manner as illustrated in Figure 6.11.

After the Gabor filtering stage the network has changed each centre-surround representation into a collection of oriented bar versions of the image. The total representation count is now increased but the activity should more linearly separable. Figure 6.12 shows a reconstruction from the spike activity for each filter.

At the final stage of the classifier network we add one population per required class. Each neuron will connect to the Gabor-filtered version of the inputs with a fixed probability of 50%; this was done to induce neurons to learn different representations of the same class.

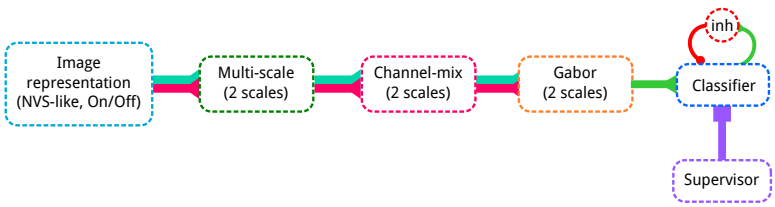
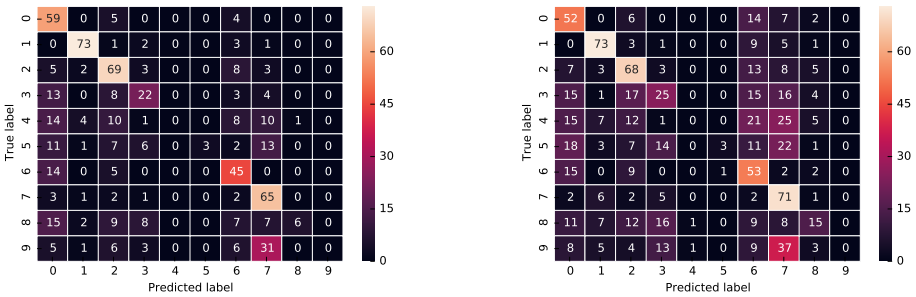


Figure 6.13: Full pipeline for the digit classifier.

Within each population neurons compete for representations through a SWTA circuit. Since this is a supervised approach, during the training stage, there is no competition between populations which represent different digits.

To train the network we showed it 200ms spike trains per digit in the TRAINING set of the MNIST hand-written digit database; each image was converted using the NVS emulator presented in Chapter 5. Specialization between classifier neurons was achieved by providing a reward pulse to the output population which was selected to represent the input digit. Additionally, we randomly provide a punishment (negative-weighed reward) signal to populations which are not the "correct" one. We chose to give random punishments to be able to keep the learning rate at a small value. The network was exposed to the full TRAINING set 3 times.

Unfortunately, this procedure was not able to produce a network which performs with high recognition accuracy when tested against samples of TESTING set from the MNIST database. The maximum accuracy obtained was 41% with an average of 38%; interestingly, these figures do not vary much whether the criterion for correctness was the number of spikes (rate) or how early a neuron spiked (time). Figures 6.14a and 6.14b show the confusion matrices for each criterion. From these we can observe that most failures to appropriately recognise digits come from numbers 4, 5, 8 and 9.



(a) Rate (b) Time

Figure 6.14: Confusion matrices for rate and time criteria.

Weight evolution shows that network parameters aggressively oscillate as some weights go from a 100% of the maximum allowed to 0% (Figure 6.15). We believe this is the main problem with the training procedure and it is due to the interaction between reward and punishment signals. In our experiments, a combination of large rewards with minuscule punishments were required to keep the network firing.

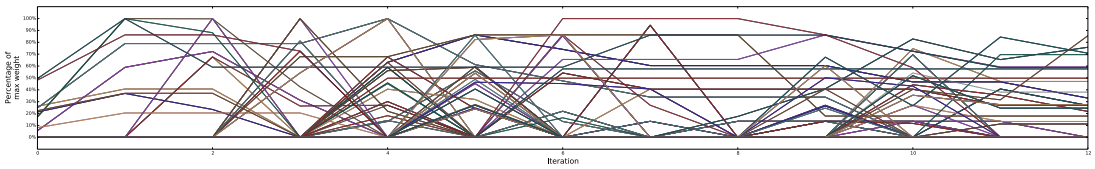


Figure 6.15: Example of weight evolution through training.

Voltage-based learning

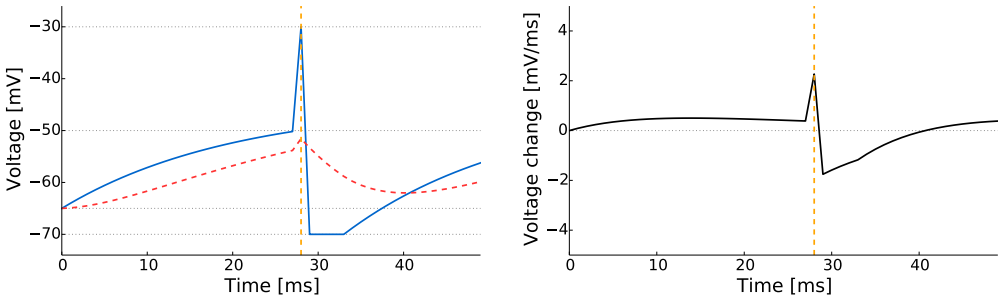
In the previous section we commented on the lack of supervision of STDP; now we focus on how it may not be a good fit for systems which cannot keep memory of $< pre, post >$ spike pairs to apply weight changes at a later time. A related problem arises from STDP's requirement of spike pairs; lack of post events means the algorithm could be missing opportunities to collect statistics.

Bengio et al. developed a learning rule which produces an STDP-like curve when weight changes are ordered with post-synaptic spike timing as the frame of reference [Bengio et al., 2015]. The same rule can be seen as optimizing the network towards predicting its next state. Scellier and Bengio extended this framework and laid a clearer link to machine learning, in particular to back-propagation [Scellier and Bengio, 2017]. Moreover, there seems to be a link between the rule and contrastive divergence, as it requires computing a stable state for the network when an input is provided and it is compared to the state of the network when both the input plus its desired class are given.

Clopath et al. proposed a voltage-based plasticity rule which mimics STDP response [Clopath et al., 2010]. This requires keeping two different low-pass filtered versions of the membrane voltage; one is used in LTP, the other in LTD. These low-pass versions of voltage are compared to a threshold and, if greater, weight changes are applied. In the case of LTP, the change is modulated by a third component which could represent NMDA or serotonin, known for facilitating learning. The magnitude for LTD is dynamic and keeps depression bound by a homeostatic mechanism.

To implement plastic synapses in hardware, Qiao et al. created a rule where weight changes depend on both membrane voltage and a slow-changing chemical level (calcium in their work) [Qiao et al., 2015]. Weight changes are non-zero when voltage is above a threshold and calcium level is within an appropriate range. Additionally weights drift exponentially to one of two constant, stable values.

In this section we describe our implementation of the rule developed by Bengio et al., our work solves noise issues O'Connor et al. [2018] by applying an on-line low-pass filtering technique. We also perform tests to establish that the compatibility with STDP of the implemented rule. An different input will be used to serve as a teaching signal, this input is an abstraction of two aspects of NMDA interaction in biological neurons. First, the current generated by this input will only be taken into account by the neuron when its membrane voltage is above a threshold, similar to how NMDA receptors in are activated in synapses. Second, the temporal dynamics of the post-synaptic potential are slow as NDMDA spikelets Antic et al. [2010]; Mel [1992]; Schiess et al. [2016].



(a) (b)

Figure 6.16: LIF neuron - DC response. a) Membrane voltage [$v(t)$; blue continuous] and its low-passed filtered version [$u(t)$; red dashed]. b) Filtered signal change in time [$\Delta u(t)/\Delta t$]

Filtered voltage change

Figure 6.16a shows the membrane voltage of a Leaky-Integrate-and-Fire (LIF) neuron when a DC current is applied. An important observation is that it changes abruptly near the spiking region (at around 30ms in the figure). In fact, in the continuous domain $dv_m/dt \rightarrow \pm\infty$, which limits the application of voltage-change-based rules. To diminish this effect, voltage (blue line in Figure 6.16a) is filtered with the exponential smoothing technique [Natrella, 2010]:

$$\gamma = e^{-1/\tau_u}, \tag{6.17}$$

$$u(t) = (1-\gamma)v_m(t) + \gamma u(t-1), \tag{6.18}$$

where u is the filtered version of membrane voltage (v_m). The constant γ dictates how much history affects the current value of u and, as a consequence, how smooth u is.

The resulting signal (red dashed line in Figure 6.16a) resembles calcium concentration in biological neurons [Senn, 2002]. It can be used to compute voltage change so that abrupt changes do not dominate the learning rule. The filtered voltage change signal ($\Delta u/\Delta t$) Figure 6.16b) for the filtered version roughly resembles STDP curves, if we set the temporal reference as the post-synaptic spike time (dashed, orange, vertical line). We could also see this as integrating spike activity over longer periods of time than the neuron dynamics can achieve, similar to what is expected with rate-coding.

Weight changes are proportional to fluctuation in the proxy (low-pass filtered) signal ($\dot{u}$)

$$\Delta w \propto f(t_{pre}, u) \tag{6.19}$$

The final weight change expression for this learning rule is

$$\Delta w = \alpha \times \delta(t - t_{pre}) \times \frac{\Delta u(t)}{\Delta t} \tag{6.20}$$

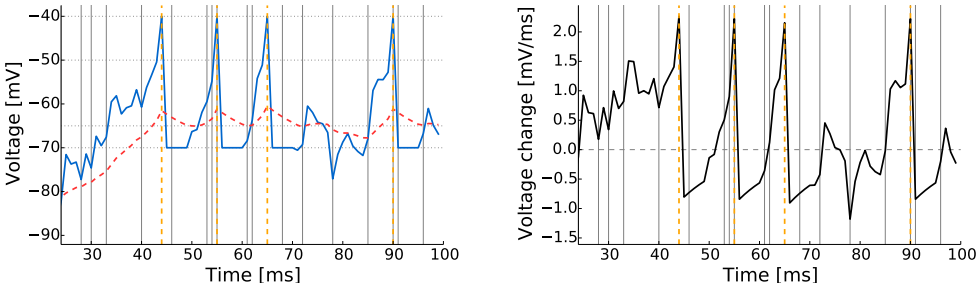
where α is a scaling factor or learn rate and δ is the Dirac delta function which limits weight changes to be generated only when a spike arrives.

To test if STDP-like behaviour is still present with our adaptation to the voltage-change rule (DVDT), we establish an experimental set-up similar to the one presented by Bengio et al. [Bengio et al., 2015]. We simulate 5000 LIF neurons using the closed-form solution (Section 2.1) programmed in the Python language. Each neuron has a noisy base current, uniformly sampled in the range $[-1, 1)nA$.

Table 6.5: LIF neuron parameters for the DVDT rule

	Threshold	Resting potential	Reset potential	Spike potential	Membrane capacitance	τ_{mem}	τ_{ref}	τ_u
Value	-50.0	-65.0	-70.0	-40.0	0.25	20.0	2.0	10.0
Units	mV	mV	mV	mV	nF	ms	ms	ms

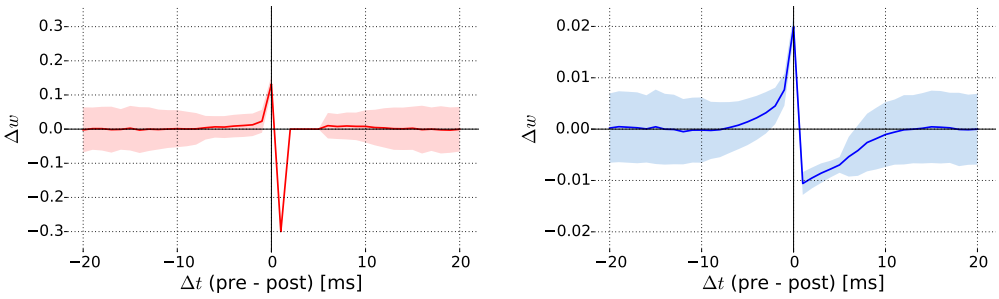
Random input spikes are generated at every time step, with a 20% and 5% probability, for excitatory and inhibitory types, respectively. All synapses are characterized as a $1ms$ pulse response; weights for inhibitory synapses are fixed, while excitatory are plastic. Voltage, voltage change and low-pass voltage change, samples are shown in Figure 6.17 where plots on the left depict membrane voltage, and right plots illustrate voltage change (black) and its filtered version (red). Orange, dashed vertical lines indicate post-synaptic neuron spikes; grey vertical lines are pre-synaptic spikes.



(a) (b)

Figure 6.17: Samples of internal state of neuron. Left membrane voltage, noisy as input spikes are also noisy. Right voltage change in black, variates violently and this makes predicting spikes hard; red is the same signal but filtered, as it does not change as much it allows us to predict easier. Also if we look at the shape of the curve near post spikes (orange vertical) it resembles STDP.

We then look for post-synaptic neuron spikes and collect weight change statistics in a $\pm 20ms$ temporal window in $1ms$ time steps. Additionally, we do the same for the non-filtered voltage changes. We compute the average change for each time step, Figure 6.18a depicts the resulting averages for voltage changes.



(a) (b)

Figure 6.18: Average weight change near post-synaptic spikes. Left is the average of weight changes produced by using non-filtered voltage change. The right plot shows the average weight change proportional to the variation of the low-pass version of the membrane voltage, we can see closer resemblance to standard STDP curve mainly due to slow changes in the input function.

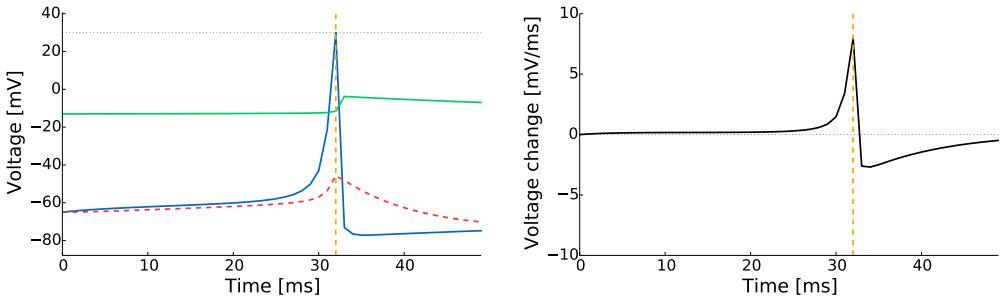
While it still looks similar to an STDP curve, the area of the negative region is quite different since it spans just a couple of milliseconds and its height is about 3 times that of the positive region; the potentiating area has a larger temporal range but changes in

weight (Δw) are smaller. In contrast, the low-pass filtered version (Figure 6.18b) has a behaviour which is closer to the STDP curve. Areas under the positive and negative curves are similar, depression is around 15% larger. Light blue shade shows the standard deviation of data; we believe the large variance is present since our neurons are constantly injected with noise.

Table 6.6: Izhikevich neuron model parameters

	a	b	c	d
Value	0.02	0.2	-65	8
Units	dimensionless			

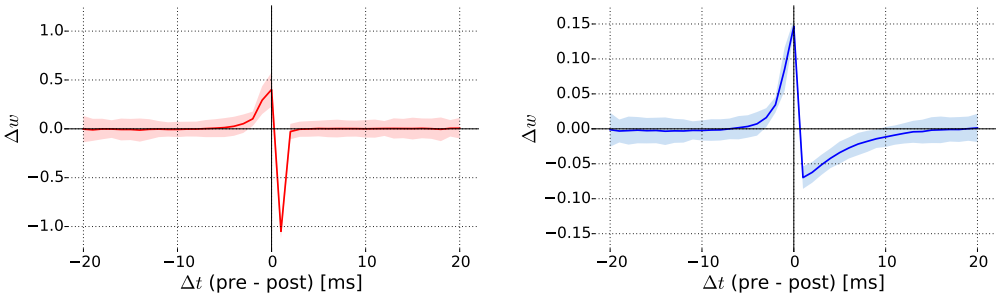
We repeated the same experiment now with Izhikevich neurons (IZK), which show a behaviour closer to biological neurons (Figure 6.19). Membrane voltage does not change as abruptly as it depends on both input current and the state of the auxiliary variable u, and the latter has really slow dynamics. This implies that the low-pass filtering does not require as wide a window as in the LIF neuron case (5ms vs 10ms).



(a) (b)

Figure 6.19: Izhikevich neuron model behaviour in response to a step input. Left - blue is the membrane voltage (v), green is the auxiliary variable u, the dashed-red is the low-pass filtered version of v. Right - black is the change in time of the low-pass filtered voltage.

While the average in the unfiltered voltage change case is closer to an STDP-like curve than in the case of LIF neurons, we still see a short temporal range(Figure 6.20a). Using the filtered version, Figure 6.20b, we observe STDP-like behaviour for the average weight change; furthermore, variance of the sample points is much smaller than in the LIF case, even with a shorter time constant τ_u.



(a) (b)

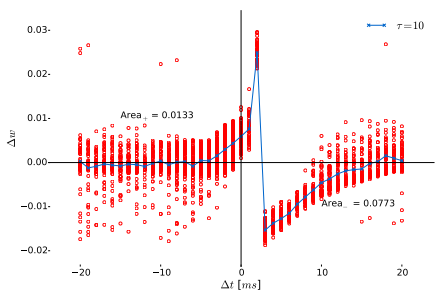
Figure 6.20: DVDT experiments with Izhikevich neurons. Left shows average weight changes using the membrane voltage derivative with respect to time. The right plot shows the average weight change when these are proportional to the temporal derivative of the low-pass filtered version of the voltage.

SpiNNaker implementation

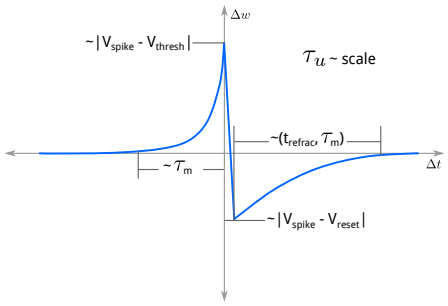
Porting the learning rule to the SpiNNaker machine required modification to the default LIF neuron model. Firstly, the current simulation model pulls the membrane voltage (v_m) to a reset level (v_{reset}) as soon as it crosses a threshold (v_{thresh}); this has the effect of increasing the LTD area. To counter this, when the threshold is crossed, we set v_m to a spike level (v_s) during the first millisecond of the refractory period, after that v_m is set to v_{reset}. A set-back for this is that a minimum $1ms$ refractory period is required.

We performed a similar experiment, as in the Python-based implementation, and computed the average of generated data points which gives rise to an STDP-like curve (Figure 6.21a). A major difference is that the curve gets shifted $1ms$ to the right; this is because weight changes are computed as soon as a spike arrives at the post-synaptic core but applied a time step after.

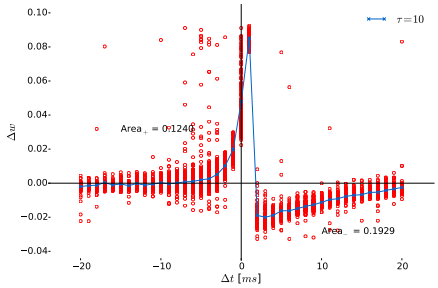
With the Izhikevich neuron model, the fast dynamics of the near threshold make the voltage value be in the range from tens to hundreds of millivolts; this means that weight increments near spikes are far greater than any decrements afterwards. In conventional computers the fast dynamics problem is usually diminished by using floating point representations and reducing the integration step. While there has been extensive work on the fixed-point solvers in SpiNNaker [Hopkins and Furber, 2015] we had to use a more primitive fix for this problem. We changed the simulation so that after the membrane voltage surpasses the threshold, the simulation also sets the membrane voltage to a (fixed) maximum value which is also just above the threshold. Changing



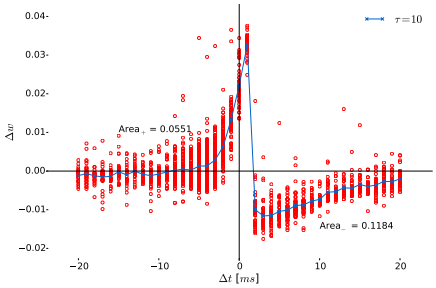
(a) LIF



(b) Curve parameters



(c) Izhikevich – standard (30mV)



(d) Izhikevich – low (-30mv)

Figure 6.21: Results of the experiments for the DVDT rule as implemented in the SpiNNaker machine.

Table 6.7: Izhikevich LTP and LTD areas for the DVDT rule

Maximum voltage (mV)	LTD area	LTP area	Ratio (LTP:LTD)
30	0.1929	0.1240	1.5556
-30	0.1184	0.0551	2.1488

this parameter scales the curve vertically but, unfortunately, also increases the ratio of LTD-to-LTP from 1.5556 to 2.1448 (Figures 6.21c and 6.21d, respectively).

To the best of our knowledge, this is the first real-time implementation of the rule on neuromorphic hardware. A great advantage over the STDP implementation is that we require fewer operations thus a 2x speed-up was observed in a small set of tests. The implementation for spiking neurons required applying low-pass filtering to the membrane voltage which also provides certain control over the shape of the LTP and LTD curves (i.e. wider moving average windows result in wider curves). An interesting aspect of the rule is that, by changing neuron parameters, the areas under the LTD

and LTP ratios change. This could mean that neurons could self-adjust the learning function depending on their spiking regime (e.g. burst- vs regular spiking).

Plastic behaviour in a soft winner-takes-all circuit

Since competition is one of the chosen motifs for neural connectivity, we compared the behaviour of plasticity using the STDP and DVDT rules when neurons are connected using a WTA network. The hypothesis of this experiment is that STDP will not be able to capture weight changes for some of the post-synaptic neurons. Because the DVDT rule does not require post-synaptic activation all weights should change. A population of 24 neurons, randomly spiking at $400Hz$, excites a target population of 3 neurons, target neurons form part of a WTA circuit (Figure 6.22).

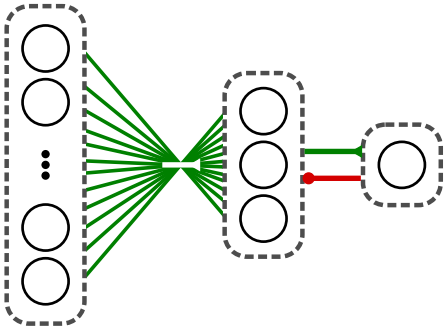
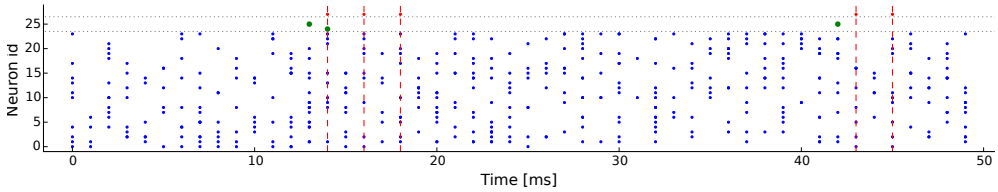


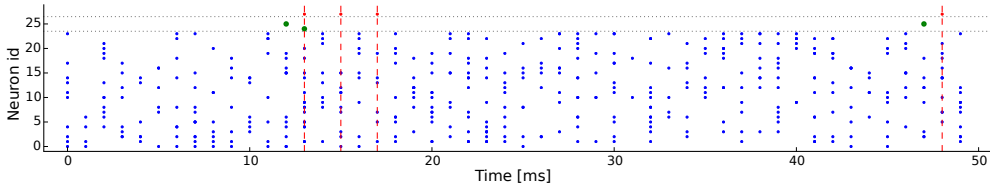
Figure 6.22: Simple WTA network. Network for the experiment; a random input consisting of 24 neurons and an all-to-all plastic connection to 3 target neurons, the latter are connected in a WTA circuit.

The experiment is run for $50ms$, where the random inputs and source to target weights are fixed for both cases (STDP and DVDT learning rules). Whenever a target neuron spikes, it is guaranteed to activate the inhibitory neuron thus blocking neighbours from spiking.

Figure 6.23 presents the spiking behaviour of the network: blue are source spikes, green are generated by target neurons and red lines indicate inhibitory neuron spikes. In both cases post neurons 1 and 2 spiked while neuron 3 did not fire. Although spiking behaviour is similar, in the case of the DVDT rule, target neurons are activated sooner, this is probably due to weight changes not requiring a post-synaptic spike.



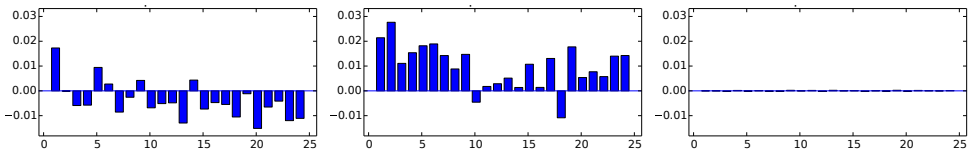
(a) STDP



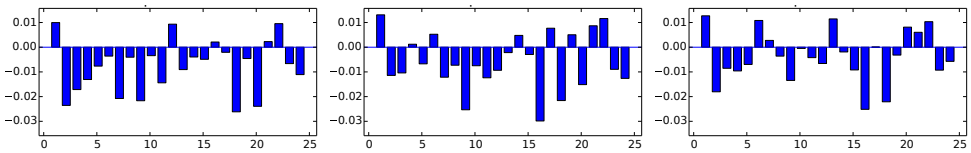
(b) DVDT

Figure 6.23: Activity for the STDP and DVDT plasticity rules; in both cases the third neuron of the target population never spiked.

Another difference is that, since post neuron 3 never spiked, weight changes are zero for the STDP case as seen in the rightmost plot in Figure 6.24a. In contrast, the DVDT rule allows for the network to adjust its weights for every pre-synaptic spike, this is reflected in rightmost of Figure 6.24b.



(a) STDP



(b) DVDT

Figure 6.24: Weight change comparison for plasticity rules. a) STDP only makes changes to the weights for the incoming connections on the first two target neurons. b) DVDT makes changes to weights of connections reaching all target neurons.

Pattern learning

One of the benefits of this rule is that it maintains compatibility with STDP which is an unsupervised learning algorithm. To test for unsupervised pattern learning we set a similar experiment as for the initial WTA test (Section 6.4.2). The main difference is that the input is now a 5×5 pixel/neuron array with two distinct noisy patterns (A, B; left side of Figure 6.25).

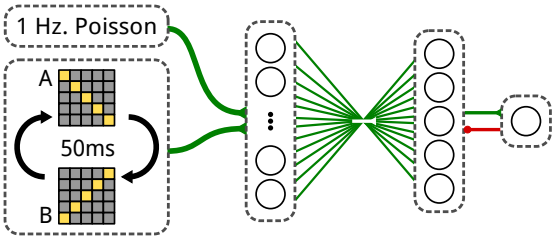
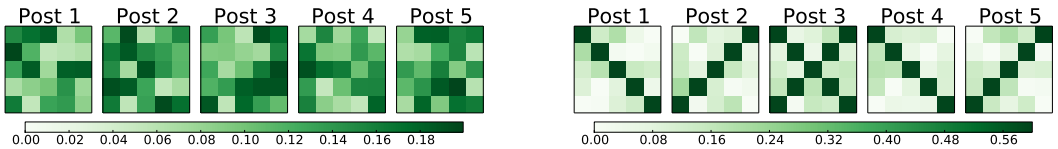


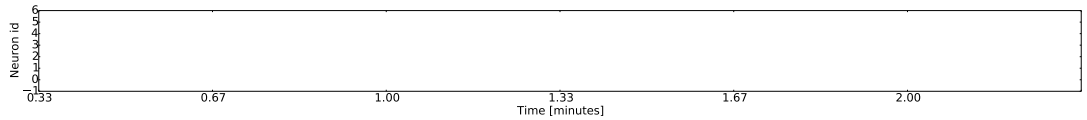
Figure 6.25: SWTA network with visual pattern as input. A 5×5 pixel/neuron array is given an input which corresponds to the two main diagonals alternated with a $50ms$ delay between them; it is also provided with a 1Hz noise with a Poisson distribution. This array is connected with plastic connections to 5 target neurons, which in turn are in a SWTA circuit.

To generate the noisy patterns we generate spikes which correspond to the two diagonals of the pixel array, these will be alternated every $50ms$. Additionally, 5×5 noise population generates spikes at $1Hz$ using a Poisson distribution. These spikes will excite the source population; every source neuron will connect to every target population neuron using plastic synapses whose initial weights are set at random using a uniform distribution with the range $(0.05, 0.2]$ (Figure 6.26a).

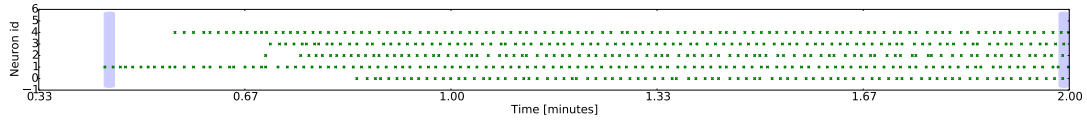


(a) (b)

Figure 6.26: Weight changes after alternating pattern simulation. a) Shows the input weights for each target neuron, these were set at random with a uniform distribution [0.05, 0.2). b) By the end of the simulation some neurons have specialized for a pattern as shown by the weights.



(a) STDP



(b) DVDT

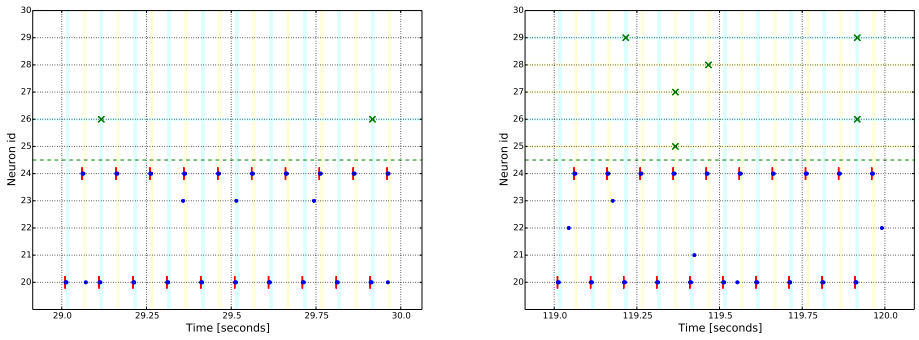
Figure 6.27: Activity during the alternating pattern input to SWTA circuit simulation. a) The simulation with the STDP algorithm never alters synaptic efficacy as the initial weights do not permit target neurons to spike. b) With the DVDT plasticity rule, after about half a minute of simulation, the target population starts to spike.

Figure 6.26b shows plastic weights at the end of the experiment. Each matrix corresponds to a post-synaptic neuron; the weight matrices have been projected to the input space. We can see that post neurons 1 and 4 specialize in pattern A, while neurons 2 and 5 do so for pattern B. Neuron 3 did not specialize, it spikes irrespective of the input.

The algorithm seems to saturate weights, for this experiment the maximum weight was 0.6nA. Notice how the efficacy of incoming synapses is close to its maximum for all output neurons. This could be a useful property as we can essentially set how many input connections are needed to make a post-synaptic neuron spike (synchronous activation). A problem with saturating weights is that, if the maximum is set arbitrarily, the SWTA circuit fails to separate input patterns (i.e. neurons spike for every pattern).

When comparing the STDP and DVDT rules, spiking behaviour shows an interesting phenomenon. While the STDP-based experiment shows no spikes (Figure 6.27a), the dynamics of the DVDT rule allows it to modify the network to a point where spikes are produced after about 30 seconds of simulation (Figure 6.27b).

Figure 6.28 shows a section of the raster plot for post-synaptic neurons in the DVDT experiment (Figure 6.27b); the initial spiking response (around the 30 second mark – Figure 6.28a) is weak.



(a) (b)

Figure 6.28: Zoom to activity of alternating pattern input to WTA circuit simulation

Unfortunately, target neurons do not spike for every presentation of the input pattern as shown in Figure 6.28b. This could be because the target neurons may still be dependent on noise or the low-pass filtering of the membrane voltage results in taking into account previous patterns. This problem can be diminished by increasing the maximum weight allowed in the experiment to the efficacy required for a neuron to spike and divide it by the number of synchronous spikes per input pattern.

Lateral signals

The notion of modulated learning is a powerful, biologically-plausible one (Section 6.3) as it can serve to provide reward or punishment signals to the learning rule. We extended the DVDT rule to have a modulator-like component; this is done by increasing the learning rate when a special signal is present. The implementation also requires a third factor, expressed as an extra receptor which does not interact with membrane voltage directly but it is kept as a second compartment. When the concentration for this transmitter $[c(t)]$ is above a certain threshold, it facilitates LTP. This is expressed in the following equation

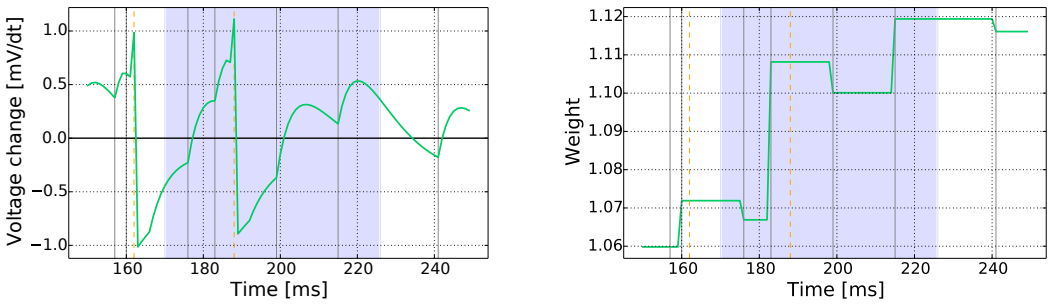
$$\Delta w = (\alpha + \beta) \times \delta(t - t_{pre}) \times u(t) \,, \tag{6.21}$$

where β is the increment factor and its value changes according to

$$\beta = \begin{cases} \beta_0 & : \quad c(t) \geq \theta_{lat} \text{ and } u(t) > 0 \,, \\ 0 & : \quad \text{otherwise} \end{cases} . \tag{6.22}$$

In the previous equation $c(t)$ is the concentration for the second compartment, and θ_{lat} is a static threshold which has to be met to activate the boost β_0. Following Hebbian principles, the voltage change proxy, $u(t)$, has to be positive.

To test the mechanism we connected single source and modulator neurons to a target one. Both the source and modulator spiked at random times. Figure 6.29a depicts signal $u(t)$, this will serve as a reference to weight evolution shown in Figure 6.29b. Boosting could be achieved in the shaded region in both plots, since during this period concentration $c(t) >= \theta_{lat}$.



(a) Early (b) Late

Figure 6.29: Effect of lateral signal (shaded region) to weight change. a) Voltage change gets affected only by standard (AMPA) input and not by the lateral signals. b) Weight changes are altered, notice larger positive alterations even if the corresponding voltage change is not as large.

The first thing to notice is how negative weight changes are smaller than positive ones in the shaded region. The increment near $160ms$ is roughly $0.01nA$ and corresponds to an u value of $\sim 0.6mv/ms$; in contrast the gain close to time $185ms$ is about $0.4nA$ but the level of u is even smaller than the first case ($\sim 0.4mv/ms$).

Soft winner-takes-all networks with lateral inputs

The main goal in this Chapter is to develop supervision mechanisms for spiking neural networks. To test whether this interaction makes any difference to the learning process, i.e. we can use it as a teaching signal, we compare the following network with and without lateral signals. For this experiment, we duplicate architecture for pattern learning and make the upper target population a notional student, while the lower one will be a teacher (Figure 6.30); the parameters for neurons in both populations are shown in Table 6.8. This interaction could be to increase the chances of different

Table 6.8: LIF parameters – lateral interaction

	V_{thresh}	V_{rest}	V_{reset}	V_{spike}	C_m	τ_{mem}	τ_{ref}	τ_{syn}	τ_u
Value	-50.0	-65.0	-70.0	-30.0	0.25	20.0	5.0	1.0	10.0
Units	mV	mV	mV	mV	nF	ms	ms	ms	ms

concept neurons of firing together or as a teaching signal coming from a higher level neuron. The experiment is set so that the student network won't specialize for any of the inputs, thus only through a teaching signal could the student develop a preference.

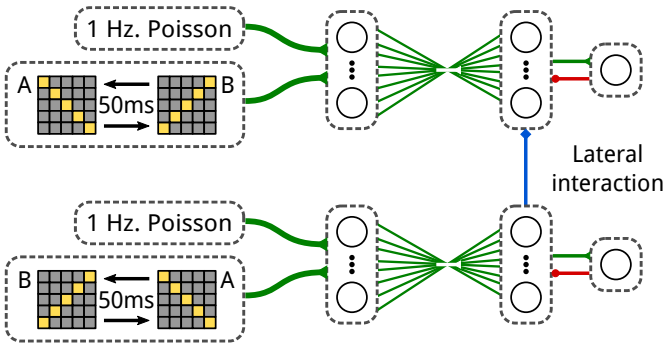
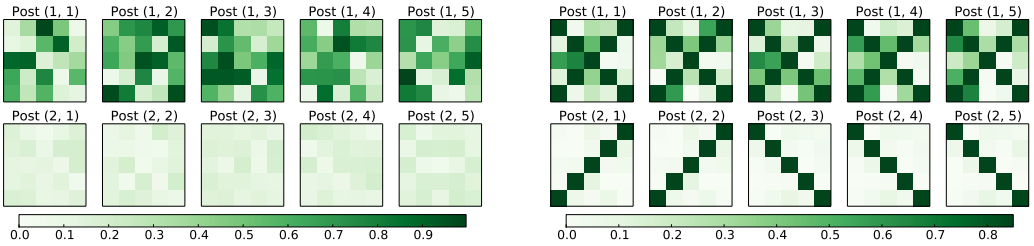


Figure 6.30: Effects of lateral inputs to WTA network. Two parallel networks as described in Section 6.4.3 with inverted diagonals. The weights for the plastic synapses of top network are initialized at a high value; the weights for the bottom network are initialized to low values. Lateral signals are sent through a one-to-one connectivity from the bottom to the top network.

The plastic weights for the student population are initially set to random values with a normal distribution whose mean is the maximum weight for the simulation. For the teacher population the weights are set low, as in the non-lateral interaction case (Section 6.4.3). We ran the experiment for 15 minutes while presenting alternating noisy patterns at $200Hz$.

For the first run we removed any lateral interaction, initial weights are shown in Figure 6.31a, the big values of the student population make it spike from the beginning of the experiment. This modifies the behaviour observed before (Section 6.4.3) and all neurons learn the weights which correspond to both incoming patterns (upper part of Figure 6.31b). The teacher population evolves as before and each neuron learns a single pattern, as seen in lower Figure 6.31b.



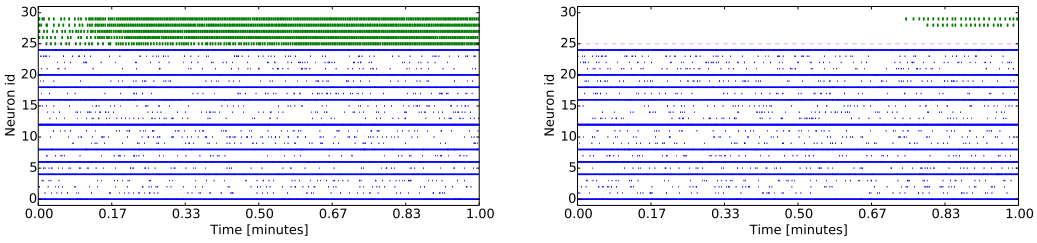
(a) (b)

Figure 6.31: Weight change for network without lateral interaction. a) Initial conditions. b) At the end of simulation.

In the next experiment we add a one-to-one connection from the teacher to the student population. To characterize the new receptor type we used additional parameters, which are shown in Table 6.9. The lateral signal from the teacher will weaken in a much longer time (about 10 times) than incoming patterns. Both the learning rate (α) and the boosting (β_0) are set to the same value (0.01), doubling the learning rate for a short period. The threshold is set to a low value to ensure lateral interaction in this experiment, though more complex networks would probably require to increase the threshold so that many neurons "supervise" the learning of a target neuron.

Table 6.9: Parameters for lateral interaction with DVDT rule

Weight	τ_{lat}	θ_{lat}	β_0
0.01	100	0.005	0.01

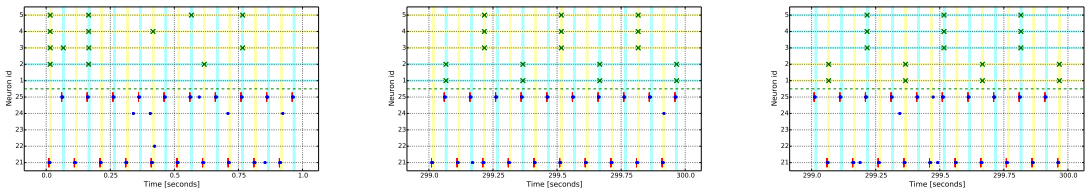


(a) Student – beginning (b) Teacher – beginning

Figure 6.32: Activity of network for student and teacher networks.

The student population begins firing early and continues to do so throughout the experiment (top of Figures 6.32a). Just before the 1 minute mark, the teacher population begins firing which, in turn, will force the student population to focus on a single pattern (top of Figure 6.32b).

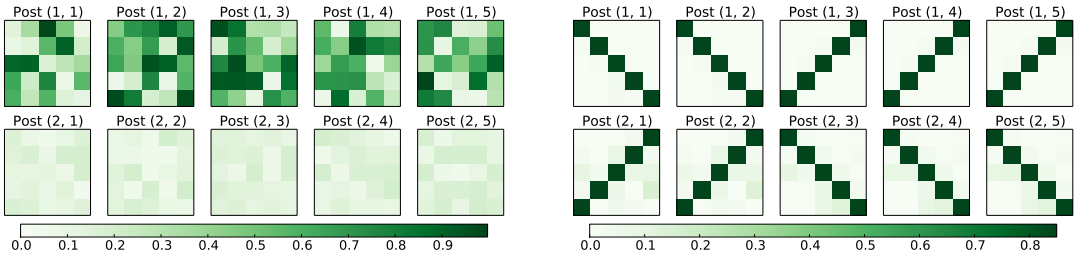
In the beginning of the simulation the student population (Figure 6.33a) spikes at random. As time progresses, both the teacher and student populations specialise and spike for a given input (Figures 6.33b and 6.33c). An interesting phenomenon is that, even though we increased the maximum weight, it seems that neurons are not activated as much. This is likely to be caused by post-synaptic neuron parameters requiring an accumulation of multiple input patterns to activate.



(a) (b) (c)

Figure 6.33: Activity of network for student and teacher networks. a) In the beginning the student population spikes at random. b) By the end neurons in the student population respond to specific patterns, dictated by the teacher population. c) Similarly to the student population, teacher neurons specialized to a particular pattern.

Finally, we show a comparison of weights at the beginning, Figure 6.34a), where student neurons have multiple synaptic weights in the $(0.8 - 1.0)$ range. At the end of the experiment, the teacher population has similar input weights as the non-lateral interaction case (bottom of Figures 6.34b and 6.31b, respectively).



(a) (b)

Figure 6.34: Changes to synaptic efficacy for WTA network with lateral interaction. Top matrices represent weights which belong to the student population, the bottom ones to the teacher neurons. a) Initial conditions. b) Weights at the end of the simulation.

The lateral interaction will encourage neurons from the student population to learn a particular spatial pattern. In this experiment we set the input activity to have opposite diagonals shown at the same time.

Biologically-inspired approach

The previous mechanism is a functional interpretation of NMDA interaction in the sense that an NMDA spike may be associated with synaptic potentiation [Ahmad and Hawkins, 2015; Antic et al., 2010]. In this section we investigate if an additional input (ϕ) with similar slow temporal behaviour can still produce this increased potentiation without contrived mechanisms. The signal will also have a similar constraint as NMDA receptors, the receiving neuron will allow current influx only when its membrane voltage is above a certain level [Mel, 1992]. Implementing this behaviour on SpiNNaker required altering how the total input current (I) is computed, by default,

$$I = \sum I_+ - \sum I_- \tag{6.23}$$

where I is the total input current given to the neuron which consists of I_+, the excitatory inputs, and I_- the inhibitory ones. With this scheme it is not possible to condition the activation of the ϕ input.

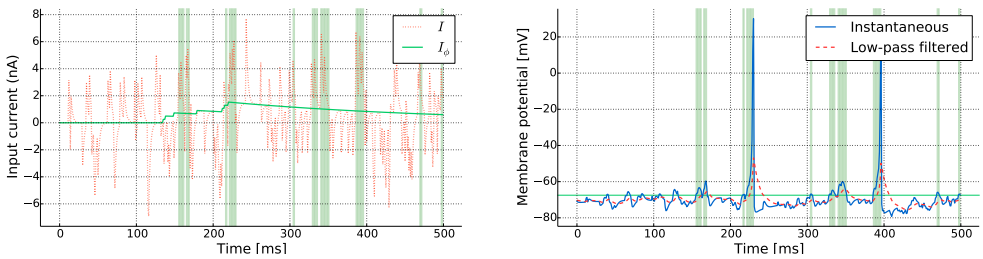
$$I = \sum I_+ - \sum I_- + \sum I_\phi \tag{6.24}$$

where I_ϕ is the current which would be provided by the ϕ input. Note that I_ϕ could further be split into positive and negative (slow) currents but has been kept positive here for simplicity. The approach taken is to simply allow current to pass when the membrane voltage is above the threshold V_ϕ:

$$I_\phi = \begin{cases} I_\phi & \text{if } V_m > V_\phi \text{ or } t - t_\phi < T_\phi \\ 0 & \text{otherwise} \end{cases} \tag{6.25}$$

We added a mechanism ($t - t_\phi < T_\phi$ in equation 6.25) which keeps the ϕ channel open for at least T_ϕ simulation steps. To achieve this we subtract the time at which an ϕ spike was last received (t_ϕ) from the current simulation time and compare this to the minimum time the channel should be open T_ϕ. This was done to allow sufficient current from the ϕ input to go into the neuron and further push the membrane voltage above the desired level (V_ϕ).

Shaded regions in Figure 6.35 show times at which ϕ is allowed to provide excitatory input to the neuron (above $V_\phi = -67.5mV$). Both I_ϕ and the total input I are shown on the left plot; notice how I_ϕ decays at a much slower rate ($\tau_\phi = 300ms$), this adds an almost-constant increment to the input current. This is subsequently reflected



(a) (b)

Figure 6.35: Reaction of input current and membrane voltage to lateral current. Shaded regions depict times at which is NMDA-enabled. a) Total input and ϕ-driven current. b) Membrane voltage and its low-pass-filtered version

in the voltage, making it likelier for spikes to occur when current from the ϕ input is present (Figure 6.35b).

Since we previously fixed lateral signals to trigger weight increments (Equation 6.21), we wanted to compare whether the influence of I_ϕ could also be perceived as a weight increment boost. To test this we set a simple network (Figure 6.36) where the source, noise and lateral populations fire at random with a 20, 40, 40 Hz frequency, respectively. The lateral population is allowed to spike only for a quarter (125*ms*) of the total simulation time (500*ms*), this to resemble multiple lateral connections spiking at about the same time. The target population consists of an Izhikevich neuron with the parameters expressed in Table 6.6.

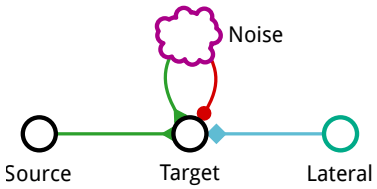


Figure 6.36: Simple ϕ experiment setup

We tested 3 different synaptic efficacy values for the lateral connection (1, 0.5 and 0.25) and repeated the experiment 20 times for each one; Table 6.10 summarizes the results for the experiments.

On average, weight changes which occurred in ϕ-inactive regions (OFF) are negative because once the target neuron spikes, it is likelier for source spikes to arrive when the voltage gradient is negative as well. Conversely, whenever a synapse efficacy is

Table 6.10: Characteristics of ϕ interaction

Lateral weight	Average weight change		Spike generation	
	ϕ OFF	ϕ ON	ϕ OFF	ϕ ON
1.0	-2.0562	1.5258	14.28%	85.72%
0.5	-1.1212	1.3915	9.52%	90.48%
0.25	-1.3462	0.5988	9.09%	90.91%

modified in ϕ active regions (ON) this change is usually positive ($\sim 98\%$). This happens because the additional, almost-constant, current input provided by ϕ activation induces an prolonged positive change on the membrane voltage. Since more current is being injected to the target neuron it is also likelier to spike. In fact, throughout the experiments, most target spikes are generated in ϕ-active regions. We can see that, as we decrease the lateral weight, positive weights are smaller as the magnitude of the injected current is also small. An advantage to having low lateral weights is that we can ensure that only with coincident source and ϕ spikes will the target neuron spike.

Pattern learning via ϕ supervision

To test whether this mechanism could work in a supervision scenario similar to the one presented in Section 6.4.4, we performed a similar experiment although a guiding signal was added to the bottom target population (lower right of Figure 6.30) and Izhikevich neurons were used instead of the LIF model. The parameters for the excitatory and inhibitory neurons are shown in Table 6.11.

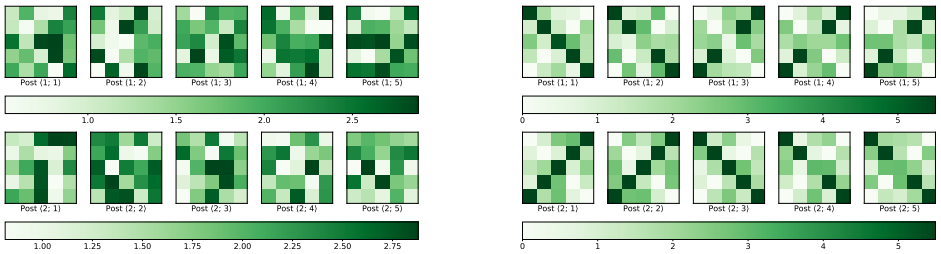
Table 6.11: Neuron parameters for pattern learning with ϕ signal.

	a	b	c	d	τ_u	τ_{exc}	τ_{inh}	τ_ϕ	θ_ϕ
Excitatory	0.01	0.2	-65	8	10	2	2	50	-75
Inhibitory	0.2	0.26	-65	0	–	1	1	–	–
Units	dimensionless				ms	ms	ms	ms	mV

The first two neurons were assigned one pattern (slash), last two neurons were set to learn the other pattern (backslash) and the remaining neuron was not assigned a target. To guide neurons in the bottom target population to learn a particular pattern we use lateral, ϕ signals which arrive 5ms before the feed-forward pattern. This time

difference can change and will (mostly) depend on the chosen synapse shape and its temporal constant. Since we chose an exponentially-decaying shape and a 50ms time constant we have a window of about 20ms for a single ϕ spike to effectively guide learning.

Similarly to the experiment in Section 6.4.4, the ϕ threshold (V_ϕ) was kept low so that ϕ is effective every time it is presented. Lateral interaction between the bottom and top target populations was kept as before, though the feed-forward pattern for the top population was set to arrive 5ms after a potential lateral spike would reach. The guiding signal was provided during a training period (29.66min) but removed for a testing phase (20s).



(a) (b)

Figure 6.37: Weights at start and end of training using an ϕ signal. a) Synaptic efficacies are set to random values initially. b) After $\sim$30min of simulation the weights favour the assigned input patterns.

Synaptic efficacy was initially set to uniformly-distributed, random values (Figure 6.37a) which, when combined, could make post-synaptic neurons spike. By the end of the experiment, input weights resemble the assigned input pattern as depicted in Figure 6.37b.

Interestingly, weights initially decrease enough to stop post neurons from spiking; after some guiding signals have been received in the bottom population weights begin to increase (Figure 6.38). The values for input synaptic efficacies corresponding to the bottom population begin to settle after about 10min. This in turn stabilises spiking activity which guide the learning process for the top population, as shown by rising lines in the top plot of Figure 6.38.

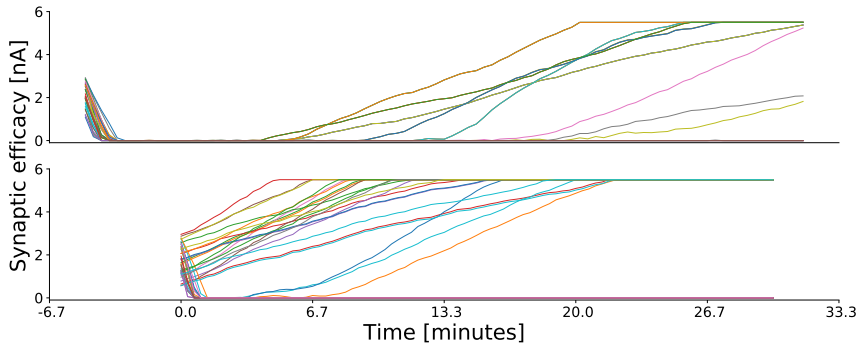
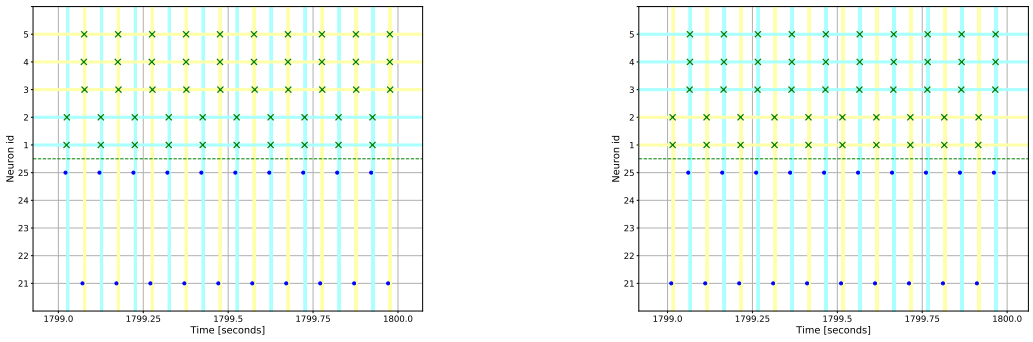


Figure 6.38: Weight evolution during training with ϕ signal.

Figure 6.39 shows the activity of the top and bottom populations for the last second of simulation. The guiding signal for the bottom population has been turned off for 19s but the learning mechanism is still active. The maximum weight was set so that a combination of 5 incoming spikes are needed to make a post-synaptic neuron to spike. Guided neurons in the bottom population (Figure 6.39b) react only to their assigned input pattern; similarly, the unconstrained neuron reacts to a single input pattern. Neurons in the top population react to specific patterns which were selected by the lateral interaction with the bottom population (Figure 6.39a).



(a) (b)

Figure 6.39: Spike activity after training with ϕ signal. The green crosses at the top represent the output for a given post-synaptic neuron. The blue dots indicate a pattern type: bottom represents the slash-like and the top illustrates the backslash one. a) shows the activity for the top population and b) spike responses for the bottom population.

Summary

While static networks have proven useful to process visual information, plasticity allows for more complex behaviours without the need to hand-craft features. Unsupervised learning has been successfully applied as the primary plasticity algorithm in SNNs. In this chapter we have described and utilized learning rules which can use an additional signal to trigger or enhance weight changes. This could allow another region of the neural network to determine whether particular synapses should be modified.

Dopamine-based reinforcement is a powerful mechanism for evaluating the performance of a network after it performed a task. Unfortunately, this learning paradigm requires several passes of the experiment to obtain good performance of the system. This happens because rewards are not provided often and punishments can overtake the learning process. In our experiments we were able to enhance efficacy for a set of neurons in an experiment similar to Pavlovian conditioning. We also demonstrated that providing this signal to neurons with distance-dependent connectivity could result in specialized receptive fields. A low classification performance was observed when training a network to recognise hand-written digits using rewards and punishments.

We have also demonstrated an STDP-compatible learning rule (DVDT) which does not require saving post-synaptic spike times. A key feature for this rule is that it allows incoming spikes (excitatory or inhibitory) to determine the direction of weight changes; furthermore, the magnitude of the synaptic change could be expressed as spiking rate. A significant advantage of the DVDT rule with respect to STDP is that weight changes occur even if the post-synaptic neuron has not spiked, thus allowing the learning algorithm to capture more of the network dynamics. We extended the DVDT algorithm to include an additional signal to trigger further weight increments. This was successfully used to train networks to recognise visual patterns and influence a neighbouring neuron to tune for a particular input. We also explored the possibility of using NMDA-like signals to influence neurons and discovered that, since an almost-constant current injection is given, weight changes are mostly positive during the presence of the additional neurotransmitter.

Chapter 7

Conclusions

We have developed the building blocks of a spike-based computer vision pipeline adopting biological principles. Engineering a vision system which resembles its biological counterpart is a difficult task; if this is done for a real-time system, difficulties increase. We now present conclusions on the topics discussed in this document.

Loading networks

Since our vision pipeline is software defined we required the network topology to be uploaded to the SpiNNaker machine; depending on the complexity of the network, this could consume more time than the actual simulation. The main problem preventing fast deployment of SNNs to the SpiNNaker machine was that an explicit representation of networks requires a lot of data to be transmitted. The problem was diminished by using a high-level description of the network, sending it to the SpiNNaker machine and allowing its computing elements to expand this description. We measured performance with different networks and the largest used was a multi-scale representation network with an input of 128×128 pixel size. The corresponding input population (representing an NVS) had a total of 32K neurons which spiked at 10Hz per neuron. We converted the input to a multi-scale representation with four scales of decreasing resolution and a total of 140K neurons. The connectivity required 3.6M synapses internally and the output population consisted of 68K ganglion cells. Loading the network when the connectivity was generated on host took about 2 hours; while on-SpiNNaker expansion resulted in a 40 minutes load time. Most time (30 minutes) is spent inspecting the network on host and transferring descriptions to the SpiNNaker machine; the time to expand the network on SpiNNaker was 10 minutes. The time to inspect

network connectivity could further be diminished if the implementation of the SpiNNaker software toolchain supported parallel computation. While the implementation presented here is not the one which ended in the main SpiNNaker toolchain, we believe the statistics showed the developers that the improvement was worth the effort.

Image acquisition

Considerable energy and time is consumed in transferring images from sensors to computing devices. Biology has many examples where image pre-processing is done as close to the sensor as possible. This is likely to avoid transferring a full image over relatively long distances. The NVS emulator described in this thesis follows this principle by reporting large changes in brightness. This emulator is open-source and available online[1]; it has been forked 8 times and gained another developer.

In the model described in Section 5.1.2 we combined two strategies used by nature: change the representation to an over-complete set of vector basis, and emit signals only when there is sufficient change in the input image. Unfortunately, changing the representation increases the output neuron count. Nevertheless, even if the neuron count is increased by about 3 times and, since we require around 10% of the neurons to spike per micro-saccade, this results in about 30% of the number of active neurons as in the original input ($0.3 \times width \times height$ neurons) but with a multi-scale representation. However, in practice we observed that active output neurons roughly accounts for $\sim$10% of the original image size or $\sim$3.5% of the output neuron count. An advantage of changing to a multi-scale representation is that low-resolution versions of the input do not vary as much over time, which could allow learning algorithms to use these nearly-constant signals to drive learning. Furthermore, low-resolution versions of the input have been typically used for localization of objects while high resolution tend to be used for recognition tasks.

Feature extraction

Common pre-processing tasks for computer vision include image filtering, motion detection and feature extraction, to name a few. These can be performed via convolution-like operations and, in our opinion, should be carried out as close to the sensor as

[1] `https://github.com/chanokin/pydvs`

possible to avoid bandwidth penalties. While a standard convolution returns a continuous number, approximating this operation in SNNs results in outputting only spikes. The number of spikes can be seen as an approximation to the continuous value for the standard version, nevertheless, each spike represents a whole region of an image. This reduces the traffic and seems a probable strategy in biological neural networks. Particularly, the input for the networks inspired by retinal circuitry developed in this thesis can be thought of as rate-coded values representing single pixels; while in the middle and output layers a single spike represents regions of the input.

In static networks we can compute the degree of inhibition neurons should provide to their neighbours by computing the cross-correlation of their input connectivity. We used this mechanism to create the multi-scale network shown in Section 5.2.1 to generate the lateral inhibition kernels. This reduces the chances of output spikes sharing the same information (i.e. spikes represent different regions or shapes).

Spiking neural networks are flexible enough to perform vision operations on static input (images) and moving stimuli (video). Temporal characteristics of synapses allow them to act as short-term decoders of incoming spike sequences; neurons integrate these decoded signals to identify features (e.g. oriented bars, centre-surround motifs) in small image regions. If synapses' time constants are set to a single time step of simulation (similar to a Dirac delta) then an instantaneous measurement is done but, when these time constants are increased, an exponential average is computed. Furthermore, it could be that conductance-based synapse models act as decoders of temporally-encoded values; in particular a scheme where the most important spikes arrive sooner than less important ones. This could happen since the first spike to arrive would have full conductance and subsequent ones would be able to access reduced channel conductivity only.

Motion is an important signal for survival as it may be a cue to avoid immediate harm or even predict long-term movement. Additionally, generative processes in the cortex (feed-back) could use these motion cues to activate or facilitate spiking of neurons even if feed-forward input is occluded. We presented models for motion sensing which are based on biological detectors. While our direction-selective circuit only senses apparent motion, neurons with a wider receptive field (e.g. at a higher level of a hierarchy) should be able to detect real motion.

Orientation filtering done with non-circular Gaussians (and its centre-surround circuit) are comparable to proposed orientation sensitive circuits in the LGN which combine the input of multiple aligned neurons with symmetric centre-surround receptive

fields.

Event-based computation could be more efficient in computer vision-related tasks such as convolutions as cores can be set to a low-power state when no input is received. Measurements taken for the static region of our visual pipeline (Section 5.4) show that an important reduction in activity can be achieved. In the SpiNNaker machine, this is tied to population partitioning and placement which is currently performed as a 1D array and is not a good fit for image processing.

Plastic networks

Computer vision systems aim to understand visual input and, possibly, decide actions based on this understanding. Achieving some form of understanding often requires a mechanism to capture the statistics of the input. In general, ANNs use changes to connections' weights to build a probabilistic graphical model of the input statistics. Most plasticity algorithms for SNNs follow Hebbian-like strategies, STDP being the most widely used. Unfortunately, STDP is an unsupervised learning procedure so modifying the weights in the network for it to perform a specific task requires modification to the STDP rule. A biologically-plausible change is to add modulation to synaptic weight changes, in particular, using a dopamine-based model.

Reinforcement learning

We tested a modulated STDP plasticity algorithm implemented for the SpiNNaker machine and which can be used for reinforcement learning. Initial testing was done with an experiment where a group of neurons was selected as the preferred input and synapses were reinforced whenever these neurons were active. After multiple presentations of the input pattern and the reinforcement signal, the network shows a high response rate for the specified pattern. This experiment was performed with biologically-plausible time constants for neurons and synapses; however, altering these can result in a viable algorithm for supervised learning.

Given the multi-scale representation and distance-dependent connectivity we obtained receptive fields which are similar to Gabor filters. The weights in these resemble the input distribution more than standard Gabor filters. Since the input is a multi-scale

representation we observed a stabilization of weights originating from lower resolutions which, in turn, helps the synapses coming from the high resolution representation to stabilize, even in the presence of noise. An undesired effect of using low-resolution versions is that it renders the input pattern indistinguishable; for example, using MNIST digits as inputs, a 5 looks practically the same as a 6 in low resolutions. This could probably be resolved by giving different importance to these resolutions; perhaps the low-resolution version provides a hint and high-resolution input delivers the decisive factor for the post-synaptic neuron to activate.

There are multiple benefits of modulating plasticity in neural networks. Firstly, a modulator can also be used as a plasticity enabler, providing stability to learned structures; stopping modulator signals would keep the network in the same state. Additionally, having control over the direction of the computed weight change can result, not only in positive reinforcement, but a negative one also which could reduce the importance synapses have to a particular input. Another case would be to bind representations in nearby areas; for example, neurons from neighbouring regions could co-supervise each other. If we think of higher cortical layers we could bind different representations of complex concepts together (e.g. spoken and written words).

A weakness of reinforcement learning is that, typically, there are many more punishments than rewards. This could, potentially, reduce network activity to a point where learning is essentially disabled. Since this framework is based on STDP, even if dopamine is present but the $<$ pre, post $>$ neurons did not spike, no plasticity would be triggered. We found that making the reinforcement for rewards much stronger than that for punishments keeps neurons in the network spiking. Unfortunately, this creates instability in the learning process as it, usually, brings weight changes to an oscillatory state. If the network is still responding, due to noise or other exploration mechanism, learning the correct set of parameters is likely to take much longer than in conventional training algorithms. The reinforcement signal transmits only an activation for plasticity plus the direction of the change; it requires an additional agent to judge whether the activity of the neurons was expected. This could provide an approximation of an error signal encoded as spike rate.

Voltage-change-based learning

The STDP plasticity algorithm has been used in multiple studies and, although successful, it has some drawbacks. The foundation of this rule is that, weights are altered proportionally to pre- and post-synaptic neuron spike times. This could be obscuring

other neural interactions which could also modify the way synaptic efficacy changes (e.g. spike triplets).

The voltage-change plasticity rule (DVDT) developed in this thesis is based on work which intends to link machine learning to neuroscience. The change in synaptic efficacy is proportional to the voltage variation with respect to time when a spike arrives to a neuron. Since SNNs are typically noisy systems the original rule was unstable; to solve this we apply a low-pass filter to the membrane voltage. An important aspect of this algorithm is that it allows capture of more of the network dynamics. For example, if a neuron is receiving excitatory input from multiple sources, it is likely that this neuron should be spiking. Using STDP as the plasticity adjustment algorithm, if the post neuron never spiked, all this activity is lost; in contrast, the DVDT rule would capture the increment in membrane potential and increase the efficacy. A subtlety found in our experiments is that, in some tests, more than one continuous pre spike was needed per synapse to ensure stable weight changes. This is due to the nature of the learning rule which is based on rate-coding; additionally, the small temporal window for the LTP versus the wideness of the LTD region pushes weights towards zero values.

Winner-takes-all (in both hard and soft versions) circuits have been proposed as unsupervised classification mechanisms. When a neuron "wins" it prevents others from spiking via inhibitory spikes which occlude network dynamics from STDP. If learning is done through the DVDT algorithm, the inhibitory input will not only prevent "losing" neurons from spiking but, also, drive the membrane voltage gradient to a negative value and, thus, it is likelier that input weights would decrease for these neurons. Recurrent connectivity using this algorithm should be treated with care as it could lead to a positive feedback loop since a voltage increment will result in higher weight values which, in turn, make neurons spike more and artificially inject more energy into the network.

We presented a supervised version of the algorithm where a spike received by a special receptor type becomes a weight increase signal. The additional increment will only happen when this signal is above a certain value; this is similar to the behaviour observed for NMDA interaction. With this simple modification we have shown it is possible to bind patterns from lateral regions, a form of co-supervision.

Another modification within the DVDT algorithm framework was created so that interaction NMDA-like signals could be simulated. This modification consists of

adding a receptor type which is active only when a certain membrane voltage threshold is surpassed. Additionally, the temporal dynamics of this receptor are much slower than the ones for standard receptors. These two requirements essentially mean that weight increments could be facilitated given feed-forward and lateral (or feedback) activity, via standard and NMDA-like signals, respectively. Since this is a computational model, modifying the sign of the NMDA-like signal could result in weight reduction.

Finally, because this algorithm does not need to perform as many operations, it is around two times faster than the current STDP implementation for the SpiNNaker machine; this allowed us to simulate up to 200 plastic neurons per core ($\sim$30% more than STDP). Furthermore, the current STDP implementation requires storing results for exponential functions in look-up-tables (LUTs) for computation time efficiency; the DVDT algorithm does not require the these tables so it is also more efficient in terms of memory consumption. The DVDT algorithm should also be a better fit for hardware SNN simulators since it relies only on the time at which pre-synaptic spikes arrive to the neuron body.

Comparison with traditional ANNs

Biological plausibility imposes certain constraints; for example, not being able to turn neurons from excitatory to inhibitory is a crucial difference. This implies that inhibitory neurons should learn "opposing" concepts to their target excitatory one. If we partition ANN populations into a ratio of excitatory and inhibitory in SNNs to keep the same number of neurons, we may now have to rely on fully-connected networks or a way to change connections as the experiment runs.

Special signals in ANNs allow distinct forward and backward phases, the former evaluates the network and has been approximated by researchers using SNNs. The signals in the backward phase transmit errors and are not present in standard SNNs; this has prevented spiking networks from being trained using gradient-based algorithms (e.g. the back-propagation algorithm).

Since we can efficiently simulate spiking neurons, it is possible to allow higher neuron counts while maintaining sparse connectivity. A benefit of having few incoming synapses is that neurons could become selective to particular patterns just from connectivity; furthermore, N-of-M encoding could be used throughout the simulated network.

Future work

Colour encoding in retinal circuits is achieved using opposing channels: red-centre/green-surround and yellow-centre/blue-surround. How to integrate this into an NVS-like mechanism is still an open question and an interesting avenue of research. While we have shown that spiking neurons are flexible enough to do multiple vision tasks, perhaps simpler algorithms could make better use of the hardware (e.g. a simpler event-driven, real-valued convolution operator).

Assigning regions of an input image to particular neurons can be done through careful partitioning and routing of populations. In the current version of the SpiNNaker Toolchain neurons in a population are assigned identification numbers sequentially (1D), regardless of a possible structure in the population. Neuron populations are then divided into smaller groups according to these identification numbers. These partitions then connect to others by specific rules (e.g. all-to-all, one-to-one, distance-dependent). If there is some structure in a population, such as each neuron representing a pixel from an image, such a partition scheme may hinder performance. One aspect of this detriment can be found when retrieving connectivity data from SDRAM, which is stored in a sparse matrix. Using 1D enumeration and partitioning can lead to rows with few columns; this means DMA calls are not fully utilized because only the beginning of retrieved data will be useful (see left of Figure 7.1).

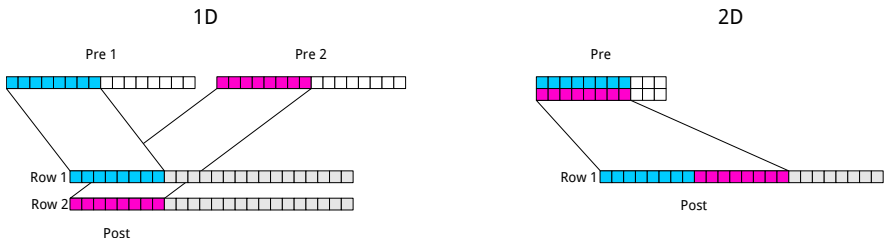


Figure 7.1: Data retrieval with 1D vs 2D populations

If we opt for 2D partitioning then a single pre-synaptic population could have more columns represented in the same synaptic row at the core simulating the post-synaptic population. This way each DMA call retrieves more useful data and efficiency increases. Additionally, routing can benefit from the structure since fewer cores would need to be connected. This could also potentially reduce routing table sizes and bottlenecks in the communication fabric.

Attention mechanisms could reduce the spatial region of input signal and, thus, the

complexity of statistics to be learned. Such a mechanism could likely act on the LGN-based channel mixing stage of our pipeline. This mechanism would require development of algorithms which mimic how the cortex understands and predicts sequences.

Reinforcement learning is key to interactive robots which could be trained by humans. Creating an actor-critic infrastructure using networks of spiking neurons is of high relevance; this would allow the simulation itself to judge whether to provide rewards or punishments (as opposed to an external entity). Furthermore, it could aid to set the rate at which these reinforcement signals are generated, thus allowing them to approximate an error signal.

The theory behind the DVDT rule suggests error signals in ANNs could be encoded as a temporal derivative of neuron state variables. Error propagation in multi-layered spiking networks is still an open question and is one of the most interesting lines of research. Work towards this goal hints at the evaluation of the network in different states or changing plasticity rules dynamically. Perhaps such changes could be implemented using different signals, such as NMDA or $GABA_B$, to propagate errors. In particular, a SpiNNaker implementation of the DVDT for any delay could allow motion detection units to be formed (similar to those presented in this thesis) or polychronous networks which are richer coincidence detectors.

Bibliography

Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., and Vijayanarasimhan, S. (2016). Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*.

Ahmad, S. and Hawkins, J. (2015). Properties of sparse distributed representations and their application to hierarchical temporal memory. *arXiv preprint arXiv:1503.07469*.

Antic, S. D., Zhou, W.-L., Moore, A. R., Short, S. M., and Ikonomu, K. D. (2010). The decade of the dendritic nmda spike. *Journal of Neuroscience Research*, 88(14):2991–3001.

Azzopardi, G. and Petkov, N. (2012). A CORF computational model of a simple cell that relies on LGN input outperforms the gabor function model. *Biological cybernetics*, 106(3):177–189.

Barlow, H. B., Hill, R. M., and Levick, W. R. (1964). Retinal ganglion cells responding selectively to direction and speed of image motion in the rabbit. *The Journal of Physiology*, 173(3):377–407.

Behnke, S. (2003). *Hierarchical neural networks for image interpretation*, volume 2766. Springer Science & Business Media.

Bengio, Y. et al. (2009). Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Bengio, Y., Lee, D., Bornschein, J., and Lin, Z. (2015). Towards biologically plausible deep learning. *CoRR*, abs/1502.04156.

Berner, R., Brandli, C., Yang, M., Liu, S.-C., and Delbruck, T. (2013). A 240×180 $10mW$ 12μs latency sparse-output vision sensor for mobile applications. In *VLSI Circuits (VLSIC), 2013 Symposium on*, pages C186–C187. IEEE.

Bhattacharya, B. S. and Furber, S. B. (2010). Biologically inspired means for rank-order encoding images: A quantitative analysis. *IEEE Transactions on Neural Networks*, 21(7):1087–1099.

Bi, G.-q. and Poo, M.-m. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 18(24):10464–10472.

Bogdan, P. A., Rowley, A. G. D., Rhodes, O., and Furber, S. B. (2018). Structural plasticity on the SpiNNaker many-core neuromorphic system. *Frontiers in Neuroscience*, 12:434.

Bohte, S. M., Kok, J. N., and La Poutré, J. A. (2000). Spikeprop: backpropagation for networks of spiking neurons. In *ESANN*, pages 419–424.

Borst, A. and Euler, T. (2011). Seeing things in motion: models, circuits, and mechanisms. *Neuron*, 71(6):974–994.

Brodal, A. (1981). Neurological anatomy. *Relation to Clinical Anatomy.*

Bromley, P. A. (2017). Products and convolutions of gaussian probability density functions. `http://www.tina-vision.net/docs/memos/2003-003.pdf`.

Brunel, N. and Van Rossum, M. C. (2007). Lapicque's 1907 paper: from frogs to integrate-and-fire. *Biological cybernetics*, 97(5-6):337–339.

Cessac, B., Paugam-Moisy, H., and Viéville, T. (2010). Overview of facts and issues about neural coding by spikes. *Journal of Physiology Paris*, 104(1-2):5–18.

Clopath, C., Büsing, L., Vasilaki, E., and Gerstner, W. (2010). Connectivity reflects coding: a model of voltage-based stdp with homeostasis. *Nature neuroscience*, 13(3):344.

Cook, P. B. and McReynolds, J. S. (1998). Lateral inhibition in the inner retina is important for spatial tuning of ganglion cells. *Nature neuroscience*, 1(8):714.

Davison, A., Brüderle, D., Eppler, J., Kremkow, J., Muller, E., Pecevski, D., Perrinet, L., and Yger, P. (2009). PyNN: a common interface for neuronal network simulators. *Frontiers in Neuroinformatics*, 2:11.

Dayan, P. and Abbott, L. F. (2001). *Theoretical neuroscience*, volume 806. Cambridge, MA: MIT Press.

Durka, P. J. (2007). Matching pursuit. *Scholarpedia*, 2(11):2288. revision #140500.

Euler, T., Haverkamp, S., Schubert, T., and Baden, T. (2014). Retinal bipolar cells: elementary building blocks of vision. *Nature Reviews Neuroscience*, 15(8):507–519.

Field, D. J. (1994). What is the goal of sensory coding? *Neural computation*, 6(4):559–601.

Fisher, S. D., Robertson, P. B., Black, M. J., Redgrave, P., Sagar, M. A., Abraham, W. C., and Reynolds, J. N. (2017). Reinforcement determines the timing dependence of cortico-striatal synaptic plasticity in vivo. *Nature communications*, 8(1):334.

FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1(6):445.

Florian, R. V. (2007). Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502.

Fox, P., Miezin, F., Allman, J., Van Essen, D., and Raichle, M. (1987). Retinotopic organization of human visual cortex mapped with positron- emission tomography. *Journal of Neuroscience*, 7(3):913–922.

Frémaux, N. and Gerstner, W. (2016). Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in neural circuits*, 9:85.

Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119 – 130.

Funke, K., Kisvarday, Z., Volgushev, M., and Wörgötter, F. (2002). Integrating anatomy and physiology of the primary visual pathway: from LGN to cortex. In van Hemmen, J. L., Cowan, J. D., and Domany, E., editors, *Models of Neural Networks IV: early vision and attention*, pages 97–182. Springer.

Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., and Brown, A. D. (2013). Overview of the SpiNNaker system architecture. *Computers, IEEE Transactions on*, 62(12):2454–2467.

Földiák, P. (1991). Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200.

Gardner, B. and Grüning, A. (2016). Supervised learning in spiking neural networks for precise temporal encoding. *PLOS ONE*, 11(8):1–28.

George, D. and Hawkins, J. (2009). Towards a mathematical theory of cortical microcircuits. *PLoS computational biology*, 5(10):e1000532.

Gerstner, W. (2016). Hebbian learning and plasticity. In Arbib, M. and Bonaiuto, J., editors, *From Neuron to Cognition Via Computational Neuroscience*, Computational Neuroscience Series. MIT Press.

Gerstner, W., Kistler, W. M., Naud, R., and Paninski, L. (2014). *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press.

Gerstner, W., Lehmann, M., Liakoni, V., Corneil, D., and Brea, J. (2018). Eligibility traces and plasticity on behavioral time scales: Experimental support of neohebbian three-factor learning rules. *arXiv preprint arXiv:1801.05219*.

Ghodrati, M., Khaligh-Razavi, S.-M., and Lehky, S. R. (2017). Towards building a more complex view of the lateral geniculate nucleus: Recent advances in understanding its role. *Progress in Neurobiology*, 156:214 – 255.

Gibbs, M. E., Hutchinson, D., and Hertz, L. (2008). Astrocytic involvement in learning and memory consolidation. *Neuroscience and Biobehavioral Reviews*, 32(5):927 – 944.

Gollisch, T. (2009). Throwing a glance at the neural code: rapid information transmission in the visual system. *HFSP journal*, 3(1):36–46.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Grill-Spector, K. and Malach, R. (2004). The human visual cortex. *Annual Review of Neuroscience*, 27(1):649–677. PMID: 15217346.

Gross, C. G. (2007). The discovery of motor cortex and its background. *Journal of the History of the Neurosciences*, 16(3):320–331. PMID: 17620195.

Halassa, M. M., Fellin, T., and Haydon, P. G. (2007). The tripartite synapse: roles for gliotransmission in health and disease. *Trends in Molecular Medicine*, 13(2):54 – 63.

Hammond, P. (1974). Cat retinal ganglion cells: size and shape of receptive field centres. *The Journal of physiology*, 242(1):99–118.

Hawkins, J. and George, D. (2006). Hierarchical temporal memory: Concepts, theory and terminology. Technical report, Numenta.

Hebb, D. O. (1949). *The organization of behavior: A neurophysiological approach*. Wiley.[JH].

Herculano-Houzel, S. (2009). The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in Human Neuroscience*, 3:31.

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

Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544.

Hopkins, M. and Furber, S. (2015). Accuracy and efficiency in fixed-point neural ode solvers. *Neural Computation*, 27(10):2148–2182. PMID: 26313605.

Hopkins, M., Pineda García, G., Bogdan, P. A., and Furber, S. B. (2018). Spiking neural networks for computer vision. *Interface Focus*, 8(4).

Hubel, D. and Wiesel, T. (1963). Shape and arrangement of columns in cat's striate cortex. *The Journal of physiology*, 165:559–568.

Hubel, D. H., Wensveen, J., and Wick, B. (1995). *Eye, brain, and vision*. Scientific American Library New York.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572.

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, 15(5):1063–1070.

Izhikevich, E. M. (2007a). *Dynamical Systems in Neuroscience*. MIT press.

Izhikevich, E. M. (2007b). Solving the distal reward problem through linkage of STDP and dopamine signaling. *BMC Neuroscience*, 8(2):S15.

Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python. [Online; accessed June-2018].

Kandel, E. R., Schwartz, J. H., Jessell, T. M., of Biochemistry, D., Jessell, M. B. T., Siegelbaum, S., and Hudspeth, A. (2000). *Principles of neural science*, volume 4. McGraw-hill New York.

Kanerva, P. (1988). *Sparse distributed memory*. MIT press.

Karklin, Y. and Simoncelli, E. P. (2011). Efficient coding of natural images with a population of noisy linear-nonlinear neurons. In *Advances in neural information processing systems*, pages 999–1007.

Kim, H., Leutenegger, S., and Davison, A. J. (2016). Real-time 3d reconstruction and 6-dof tracking with an event camera. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 349–364, Cham. Springer International Publishing.

Koopman, P. (2002). 32-bit cyclic redundancy codes for internet applications. In *Proceedings International Conference on Dependable Systems and Networks*, pages 459–468.

Krizhevsky, A., Nair, V., and Hinton, G. (2014). The CIFAR-10 dataset. `http://www.cs.toronto.edu/kriz/cifar.html`.

Krubitzer, L. A. and Seelke, A. M. H. (2012). Cortical evolution in mammals: The bane and beauty of phenotypic variability. *Proceedings of the National Academy of Sciences*, 109(Supplement 1):10647–10654.

Kuffler, S. W. (1953). Discharge patterns and functional organization of mammalian retina. *Journal of neurophysiology*, 16(1):37–68.

Kupers, R. and Ptito, M. (2004). "seeing" through the tongue: cross-modal plasticity in the congenitally blind. *International Congress Series*, 1270:79 – 84. Frontiers in Human Brain Topology. Proceedings of ISBET 2004.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

LeCun, Y., Cortes, C., and Burges, C. J. (2010). MNIST handwritten digit database. `http://yann.lecun.com/exdb/mnist`.

Liu, Q., Pineda García, G., Stromatias, E., Serrano-Gotarredona, T., and Furber, S. B. (2016). Benchmarking spike-based visual recognition: A dataset and evaluation. *Frontiers in Neuroscience*, 10:496.

Liu, S.-C. and Delbruck, T. (2010). Neuromorphic sensory systems. *Current opinion in neurobiology*, 20(3):288–295.

Markram, H., Lübke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Science*, 275(5297):213–215.

Masland, R. H. (2012). The neuronal organization of the retina. *Neuron*, 76(2):266–280.

McIntosh, L., Maheswaranathan, N., Nayebi, A., Ganguli, S., and Baccus, S. (2016). Deep learning models of the retinal response to natural scenes. In *Advances in Neural Information Processing Systems*, pages 1369–1377.

Mead, C. and Ismail, M. (2012). *Analog VLSI implementation of neural systems*, volume 80. Springer Science & Business Media.

Mead, C. A. and Mahowald, M. (1988). A silicon model of early visual processing. *Neural Networks*, 1(1):91 – 97.

Mel, B. W. (1992). Nmda-based pattern discrimination in a modeled cortical neuron. *Neural Computation*, 4(4):502–517.

Mikaitis, M., Pineda García, G., Knight, J. C., and Furber, S. B. (2018). Neuromodulated synaptic plasticity on the SpiNNaker neuromorphic system. *Frontiers in Neuroscience*, 12:105.

Misra, J. and Saha, I. (2010). Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing*, 74(1):239–255.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Natrella, M. (2010). NIST/SEMATECH e-handbook of statistical methods. `https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc431.htm`.

Nichols, E., Gardner, B., and Gruning, A. (2017). *Supervised Learning on the SpiNNaker Neuromorphic Hardware*.

O'Connor, P., Gavves, E., and Welling, M. (2018). Training a network of spiking neurons with equilibrium propagation.

Pack, C. C. and Bensmaia, S. J. (2015). Seeing and feeling motion: canonical computations in vision and touch. *PLoS biology*, 13(9):e1002271.

Pardo, F., Boluda, J. A., and Vegara, F. (2015). Selective change driven vision sensor with continuous-time logarithmic photoreceptor and winner-take-all circuit for pixel selection. *IEEE Journal of Solid-State Circuits*, 50(3):786–798.

Patterson, C., Garside, J., Painkras, E., Temple, S., Plana, L. A., Navaridas, J., Sharp, T., and Furber, S. (2012). Scalable communications for a million-core neural processing architecture. *Journal of Parallel and Distributed Computing*, 72(11):1507–1520.

Perea, G., Navarrete, M., and Araque, A. (2009). Tripartite synapses: astrocytes process and control synaptic information. *Trends in Neurosciences*, 32(8):421 – 431.

Pillow, J. W., Paninski, L., Uzzell, V. J., Simoncelli, E. P., and Chichilnisky, E. (2005). Prediction and decoding of retinal ganglion cell responses with a probabilistic spiking model. *Journal of Neuroscience*, 25(47):11003–11013.

Pineda García, G., Camilleri, P., Liu, Q., and Furber, S. (2016). PyDVS: An extensible, real-time dynamic vision sensor emulator using off-the-shelf hardware. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7.

Ponulak, F. and Kasiński, A. (2010). Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting. *Neural Computation*, 22(2):467–510. PMID: 19842989.

Portelli, G., Barrett, J. M., Hilgen, G., Masquelier, T., Maccione, A., Di Marco, S., Berdondini, L., Kornprobst, P., and Sernagor, E. (2016). Rank order coding: a retinal information decoding strategy revealed by large-scale multielectrode array retinal recordings. *eNeuro*, 3(3).

Project, S. (2018). SpiNNaker manchester at github.com. `http://spinnakermanchester.github.io`.

Qiao, N., Mostafa, H., Corradi, F., Osswald, M., Stefanini, F., Sumislawska, D., and Indiveri, G. (2015). A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in Neuroscience*, 9:141.

Quiroga, R. Q., Reddy, L., Kreiman, G., Koch, C., and Fried, I. (2005). Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102.

Rao, R. P. and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1).

Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019.

Roelfsema, P. R. and Holtmaat, A. (2018). Control of synaptic plasticity in deep cortical networks. *Nature Reviews Neuroscience*, 19(3):166.

Roelfsema, P. R. and Ooyen, A. v. (2005). Attention-gated reinforcement learning of internal representations for classification. *Neural computation*, 17(10):2176–2214.

Roska, B. and Meister, M. (2014). *The Retina Dissects the Visual Scene into Distinct Features*, chapter 13. Mit Press.

Scellier, B. and Bengio, Y. (2017). Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, 11:24.

Schiess, M., Urbanczik, R., and Senn, W. (2016). Somato-dendritic synaptic plasticity and error-backpropagation in active dendrites. *PLOS Computational Biology*, 12(2):1–18.

Schwartz, G. and Rieke, F. (2011). Nonlinear spatial encoding by retinal ganglion cells: when 1+ 1$\neq$ 2. *The Journal of general physiology*, 138(3):283–290.

Senn, W. (2002). Beyond spike timing: the role of non-linear plasticity and unreliable synapses. *Biological cybernetics*, 87(5-6):344–355.

Serrano-Gotarredona, T. and Linares-Barranco, B. (2013). A 128×128 1.5% contrast sensitivity 0.9% FPN 3μs latency $4mW$ asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. *IEEE Journal of Solid-State Circuits*, 48(3):827–838.

Serre, T., Kouh, M., Cadieu, C., Knoblich, U., Kreiman, G., and Poggio, T. (2005). A theory of object recognition: computations and circuits in the feedforward path of the ventral stream in primate visual cortex. Technical report, Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory.

Sherman, S. M. (2006). The thalamus. `http://www.scholarpedia.org/article/Thalamus`.

Sjöström, J. and Gerstner, W. (2010). Spike-timing dependent plasticity. *Scholarpedia*, 5(2):1362. revision #184913.

Sofroniew, M. V. and Vinters, H. V. (2010). Astrocytes: biology and pathology. *Acta Neuropathologica*, 119(1):7–35.

Spruyt, V. (2017). A geometric interpretation of the covariance matrix. `http://www.visiondummy.com/2014/04/geometric-interpretation-covariance-matrix/`.

Srinivasan, M. V., Laughlin, S. B., and Dubs, A. (1982). Predictive coding: a fresh view of inhibition in the retina. *Proceedings of the Royal Society of London B: Biological Sciences*, 216(1205):427–459.

Sterling, P. (2015). Beyond the retina: Pathways to perception and action. In Sterling, P. and Laughlin, S., editors, *Principles of Neural Design*, pages 323–362. MIT Press.

Stromatias, E., Galluppi, F., Patterson, C., and Furber, S. (2013). Power analysis of large-scale, real-time neural networks on spinnaker. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

Tewari, S. G. and Majumdar, K. K. (2012). A mathematical model of the tripartite synapse: astrocyte-induced synaptic plasticity. *Journal of Biological Physics*, 38(3):465–496.

Theodosis, D. T., Poulain, D. A., and Oliet, S. H. R. (2008). Activity-dependent structural and functional plasticity of astrocyte-neuron interactions. *Physiological Reviews*, 88(3):983–1008.

Thompson, R. (2000). *The Brain: A Neuroscience Primer*. Worth Publishers.

Thorpe, S., Delorme, A., and Van Rullen, R. (2001). Spike-based strategies for rapid processing. *Neural Networks*, 14(6):715 – 725.

Tieleman, T. (2008). Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM.

Tootell, R. B. and Hadjikhani, N. (2001). Where is 'Dorsal V4' in human visual cortex? retinotopic, topographic and functional evidence. *Cerebral Cortex*, 11(4):298–311.

Van Rullen, R. and Thorpe, S. J. (2001). Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural computation*, 13(6):1255–1283.

VanRullen, R., Guyonneau, R., and Thorpe, S. J. (2005). Spike times make sense. *Trends in Neurosciences*, 28(1):1 – 4.

Whittington, J. C. and Bogacz, R. (2019). Theories of error back-propagation in the brain. *Trends in Cognitive Sciences*, 23(3):235 – 250.

Wikipedia (2017a). List of AMD Graphics Processing Units. `https://en.wikipedia.org/wiki/List_of_AMD_graphics_processing_units`.

Wikipedia (2017b). List of Nvidia Graphics Processing Units. `https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units`.

Wohrer, A. and Kornprobst, P. (2009). Virtual Retina : A biological retina model and simulator, with contrast gain control. *Journal of Computational Neuroscience*, 26(2):219.

Zhang, Y., Kim, I.-J., Sanes, J. R., and Meister, M. (2012). The most numerous ganglion cell type of the mouse retina is a selective feature detector. *Proceedings of the National Academy of Sciences*, 109(36):E2391–E2398.

Appendix A

Connector descriptions

In this appendix we describe the required data to perform on-SpiNNaker connection expansion; each section describes a component of the expander. Names (e.g. One-To-One, All-To-All) are encoded using a cyclic redundancy check (CRC) and used to identify the component type (for compression purposes). The encoded name is a required parameter for each component but some require additional data which will be described in the following sections.

Connectors

We begin by listing acceptance rules for different connectivity patterns. The first three are basic connection styles found in the PyNN description language, while the last two where developed solely for this thesis.

One to one

Input: None

begin

  for $i \in pre$ **do**

    for $j \in post$ **do**

      if $i = j$ **then**

        Accept

      end

    end

  end

end

All to all

Input: *allow*: Will neurons with the same id be allowed to connect.

```
begin
    for i ∈ pre do
        for j ∈ post do
            if i = j and not allow then
                Reject
            else
                Accept
            end
        end
    end
end
```

Fixed probability

Input: *allow*: Will neurons with the same id be allowed to connect.

Input: *prob*: Probability that neuron pairs form synapses.

```
begin
    for i ∈ pre do
        for j ∈ post do
            sample ← SampleUniform(0, 1)
            if (i = j and not allow) or prob > sample then
                Reject
            else
                Accept
            end
        end
    end
end
```

Kernel

Input: $shape_p$: Shape of pre-synaptic population (2D).

Input: $offsets$, $steps$: Spatial sampling offsets and steps.

Input: $shape_k$: Shape of kernel weights

begin

  for $i \in pre$ **do**

    for $j \in post$ **do**

      $valid\_region \leftarrow$ ProjectKernel$(j, shape_p, shape_k, offsets, steps)$

      if $i \in valid\_region$ **then**

        Accept

      else

        Reject

      end

    end

  end

end

Cortical

Input: $shape_p$: Shape of pre-synaptic population (2D).

Input: $distance_{max}$: How far should neuron pairs may be to consider generating a synapse.

Input: $offsets$, $steps$: Spatial sampling offsets and steps.

Input: $allow$: Will neurons with the same id be allowed to connect.

Input: $prob$: Probability that neuron pairs form synapses.

begin

  for $i \in pre$ **do**

    for $j \in post$ **do**

      $distance \leftarrow$ ComputeDistance$(i, j, shape_p, offsets, steps)$

      $sample \leftarrow$ SampleUniform$(0, 1)$

      if *(*$i = j$ **and not** $allow$*)* **or** $prob > sample$ **or** $distance > distance_{max}$ **then**

        Reject

      else

        Accept

      end

    end

  end

end

Weights and delays

Each synapse has an associated weight (efficacy) and delay; on-SpiNNaker generation of these values requires additional parameters which are described in the following table.

Table A.1: Weight and delay generation parameters.

Type	Parameters
Constant	Value
Random uniform	Range extrema
Random normal	Mean and standard deviation
Kernel	Values

Synapse types

In SNNs simulations we can make weights remain static or the dynamics of the network to modify them. When static, the on-SpiNNaker synapse generator does not require further information as a 16-bit integer is the standard used for weights. If plasticity is enabled, synapse generators require to know how much space is needed to store additional plasticity parameters.

Table A.2: Static and plastic synapse generator parameters.

Type	Parameters
Static	None
Plastic	Number of 32-bit words to save weight state

Appendix B

Competition of Gaussian receptive fields

Convolution of two 1D Gaussian functions

Let $G_i(x)$ be a Gaussian function

$$G_i(x) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-x^2/(2\sigma_i^2)} \,, \tag{B.1}$$

with mean $\mu = 0$ and variance equal to σ_i^2. The convolution of two functions is defined by

$$f(x) \circledast g(x) = \int_{-u}^{u} f(u) \cdot g(u-x) du \,. \tag{B.2}$$

The convolution operation in the frequency domain is a multiplication [REF],

$$f(x) \circledast g(x) = F^{-1} \left[F\left[f(x)\right] \cdot F\left[g(x)\right] \right] \,, \tag{B.3}$$

where F is the Fourier transform

$$F[f(x)] = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx \,, \tag{B.4}$$

and F^{-1} is its inverse. We can compute the convolution of two 1D Gaussian functions by transforming them to the frequency domain

$$\begin{aligned} F\left[G_i(x)\right] &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-x^2/(2\sigma_i^2)} e^{-2\pi i k x} dx \,, & \text{(B.5)} \\ &= \frac{1}{\sqrt{2\pi}\sigma_i} \int_{-\infty}^{\infty} e^{-x^2/(2\sigma_i^2)} \left[cos(-2\pi k x) - i \cdot sin(-2\pi k x)\right] dx \,, & \text{(B.6)} \\ &= \frac{1}{\sqrt{2\pi}\sigma_i} \int_{-\infty}^{\infty} e^{-x^2/(2\sigma_i^2)} \left[cos(-2\pi k x)\right] dx \,. & \text{(B.7)} \end{aligned}$$

This integral has an established solution

$$\int_0^{\infty} e^{-ax^2} cos(2bx) dx = \frac{1}{2}\sqrt{\frac{\pi}{a}} e^{-(b^2/a)} \,, \qquad \text{(B.8)}$$

so the Fourier transform of G_i is

$$\begin{aligned} F\left[G_i(x)\right] &= \frac{1}{\sqrt{2\pi}\sigma_i} \cdot 2 \cdot \frac{1}{2}\sqrt{\frac{\pi}{\frac{1}{2\sigma_i^2}}}\, e^{-\frac{(\pi k)^2}{1/(2\sigma_i^2)}} & \text{(B.9)} \\ &= \frac{1}{\sqrt{2\pi}\sigma_i} \sqrt{2\pi\sigma_i^2}\, e^{-2\sigma_i^2\pi^2k^2} & \text{(B.10)} \\ &= e^{-2\sigma_i^2\pi^2k^2} \,. & \text{(B.11)} \end{aligned}$$

We can now produce the convolution through the Fourier transforms of $G_0(x)$ and $G_1(x)$

$$\begin{aligned} F^{-1}\left[F\left[G_0(x)\right] \cdot F\left[G_1(x)\right]\right] &= F^{-1}\left[e^{-2\sigma_0^2\pi^2k^2} \cdot e^{-2\sigma_1^2\pi^2k^2}\right] & \text{(B.12)} \\ &= F^{-1}\left[e^{-2\pi^2k^2(\sigma_0^2+\sigma_1^2)}\right] & \text{(B.13)} \\ &= F^{-1}\left[e^{-2\pi^2k^2(\sigma'^2)}\right] & \text{(B.14)} \\ &= \frac{1}{\sqrt{2\pi}\sigma'} e^{-\frac{x^2}{2(\sigma'^2)}} & \text{(B.15)} \\ &= \frac{1}{\sqrt{2\pi(\sigma_0^2+\sigma_1^2)}}\, e^{-\frac{x^2}{2(\sigma_0^2+\sigma_1^2)}} & \text{(B.16)} \end{aligned}$$

Convolution of two 2D Gaussian functions

A 2D Gaussian function with independent variables x and y is defined as

$$G_i(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{(-[(x-\mu_x)^2/(2\sigma_x^2)+(y-\mu_y)^2/(2\sigma_y^2)])}. \tag{B.17}$$

We will constrain 2D Gaussian functions to have

$$\mu_x = \mu_y = 0 \quad \text{and} \quad \sigma_x = \sigma_y \,,$$

The competition of two neurons with Gaussian receptive fields is given by the correlation of their input weight matrix. Since their receptive fields are based on an odd function, their correlation is equal to their convolution, which is given by

$$G_0(x,y) \circledast G_1(x,y) \;\; = \;\; \int_{-u}^{u}\int_{-v}^{v} G_0(u,v)\cdot G_1(x-u,y-v)\,du\,dv\,. \tag{B.18}$$

The convolution of these functions is separable

$$\begin{aligned}
&\int_{-u}^{u}\int_{-v}^{v} G_0(u,v)\cdot G_1(x-u,y-v)\,du\,dv = \\
&\int_{-u}^{u}\int_{-v}^{v} \frac{1}{2\pi\sigma_0^2} e^{(-[(u^2+v^2)/(2\sigma_0^2)])} \cdot \frac{1}{2\pi\sigma_1^2} e^{(-[((x-u)^2+(y-v)^2)/(2\sigma_1^2)])}\,du\,dv = \\
&\int_{-u}^{u}\int_{-v}^{v} K_0K_1 e^{(-[(u^2+v^2)/(2\sigma_0^2)])} \cdot e^{(-[((x-u)^2+(y-v)^2)/(2\sigma_1^2)])}\,du\,dv = \\
&\int_{-u}^{u} K_0K_1 \cdot e^{(-[u^2/(2\sigma_0^2)])} \cdot e^{(-[(x-u)^2/(2\sigma_1^2)])}\,du \int_{-v}^{v} e^{(-[v^2/(2\sigma_0^2)])} \cdot e^{(-[(y-v)^2/(2\sigma_1^2)])}\,dv = \\
&\int_{-u}^{u} \sqrt{K_0K_1} \cdot e^{(-[u^2/(2\sigma_0^2)])} \cdot e^{(-[(x-u)^2/(2\sigma_1^2)])}\,du \int_{-v}^{v} \sqrt{K_0K_1} \cdot e^{(-[v^2/(2\sigma_0^2)])} \cdot e^{(-[(y-v)^2/(2\sigma_1^2)])}\,dv
\end{aligned} \tag{B.19}$$

this is merely the multiplication of two 1D convolutions (horizontal and vertical axis), so the right hand side integral solutions are

$$\begin{aligned}
&\int_{-u}^{u} \sqrt{K_0K_1}\cdot e^{(-[u^2/(2\sigma_0^2)])} \cdot e^{(-[(x-u)^2/(2\sigma_1^2)])}\,du \int_{-v}^{v} \sqrt{K_0K_1} \cdot e^{(-[v^2/(2\sigma_0^2)])} \cdot e^{(-[(y-v)^2/(2\sigma_1^2)])}\,dv = \\
&\frac{1}{\sqrt{2\pi(\sigma_0^2+\sigma_1^2)}}\, e^{-x^2/[2(\sigma_0^2+\sigma_1^2)]} \cdot \frac{1}{\sqrt{2\pi(\sigma_0^2+\sigma_1^2)}}\, e^{-y^2/[2(\sigma_0^2+\sigma_1^2)]} = \\
&\frac{1}{2\pi(\sigma_0^2+\sigma_1^2)}\, e^{-(x^2+y^2)/[2(\sigma_0^2+\sigma_1^2)]}\,,
\end{aligned} \tag{B.20}$$

finally, the convolution is

$$G_0(x,y) \circledast G_1(x,y) = \frac{1}{2\pi(\sigma_0^2+\sigma_1^2)}\, e^{-(x^2+y^2)/[2(\sigma_0^2+\sigma_1^2)]}\,. \tag{B.21}$$

Appendix C

Event-driven reinforcement learning rule

Izhikevich [2007b] describes modulated STDP model with the following equations (Section 6.3).

$$\frac{dc(t)}{dt} = -\frac{c(t)}{\tau_c} + STDP(\Delta t)\delta(t - t_{pre/post}) \quad \text{(C.1)}$$

$$\frac{dm(t)}{dt} = -\frac{m(t)}{\tau_m} + M(t)\delta(t - t_{mod}) \quad \text{(C.2)}$$

$$\frac{dw(t)}{dt} = m(t) \times c(t) \quad \text{(C.3)}$$

where c, m are the eligibility trace and modulator, respectively. Function $STDP(t)$ comes from Spike-timing-dependent plasticity rule; and $M(t)$ is the concentration of modulator when a reward -or punishment- is received. The change in weight is characterized by the multiplication of the modulator and eligibility trace states.

We require a separation of continuous and discrete portions of Equations C.1, C.2 and C.3. This due to the event-driven computation nature of the SpiNNaker machine. To know what the current level of variables c and m we integrate their equations with time limits set to their last update. Functions $STDP(t)$ and $M(t)$ are not continuous (i.e. it's an instantaneous change) so integration is not required.

Both $dc(t)/dt$ and $dm(t)/dt$ are linear differential equations and integration can be

done as follows:

$$\frac{dx(t)}{dt} = -\frac{x(t)}{\tau_x} \tag{C.4}$$

$$\frac{dx(t)}{x(t)} = -\frac{1}{\tau_x}dt \tag{C.5}$$

$$\int_{t_{last}}^{t} \frac{dx(t)}{x(t)} = -\int_{t_{last}}^{t} \frac{1}{\tau_x}dt \tag{C.6}$$

$$\ln x(t) - \ln x(t_{last}) = -\frac{t - t_{last}}{\tau_x} \tag{C.7}$$

$$\ln \frac{x(t)}{x(t_{last})} = -\frac{t - t_{last}}{\tau_x} \tag{C.8}$$

$$\frac{x(t)}{x(t_{last})} = e^{-\frac{t-t_{last}}{\tau_x}} \tag{C.9}$$

$$x(t) = x(t_{last})e^{-\frac{t-t_{last}}{\tau_x}} \tag{C.10}$$

exponentially decaying the value since the last update.

For the weight update $[dw(t)/dt]$ integration can be achieved via

$$\frac{dw}{dt} = c(t) \times m(t) = \left(c(t_{lc})e^{-\left(\frac{t-t_{lc}}{\tau_c}\right)}\right)\left(m(t_{lm})e^{-\left(\frac{t-t_{lm}}{\tau_m}\right)}\right) \tag{C.11}$$

$$\int_{t_{lw}}^{t} dw = c(t_{lc})m(t_{lm}) \int_{t_{lw}}^{t} e^{-\left(\frac{t-t_{lc}}{\tau_c}\right)} e^{-\left(\frac{t-t_{lm}}{\tau_m}\right)} dt \tag{C.12}$$

$$\int_{t_{lw}}^{t} dw = c(t_{lc})m(t_{lm}) \int_{t_{lw}}^{t} e^{-\left(\frac{t-t_{lc}}{\tau_c} + \frac{t-t_{lm}}{\tau_m}\right)} dt \tag{C.13}$$

where le, lm and lw subscripts indicate the time at which the last eligibility trace, modulator and weight updates where performed, respectively. To solve Equation C.13 we will transform it into a linear differential equation form, thus we require variable change

$$u = -\left(\frac{t - t_{lc}}{\tau_c}\right) - \left(\frac{t - t_{lm}}{\tau_m}\right) \tag{C.14}$$

$$\frac{du}{dt} = -\frac{1}{\tau_c} - \frac{1}{\tau_m} \tag{C.15}$$

$$dt = du/\left(-\frac{1}{\tau_c} - \frac{1}{\tau_m}\right) \tag{C.16}$$

Now we can solve the next equation instead

$$\int_{t_{lw}}^{t} \frac{e^u}{-\frac{1}{\tau_c}-\frac{1}{\tau_m}} du = \left[\frac{e^u}{-\frac{1}{\tau_c}-\frac{1}{\tau_m}}\right]_{t_{lw}}^{t} \tag{C.17}$$

$$\int_{t_{lw}}^{t} e^{-\left(\frac{t-t_{lc}}{\tau_c}\right)-\left(\frac{t-t_{lm}}{\tau_m}\right)} dt = \left[\frac{e^{-\left(\frac{t-t_{lc}}{\tau_c}\right)-\left(\frac{t-t_{lm}}{\tau_m}\right)}}{-\frac{1}{\tau_c}-\frac{1}{\tau_m}}\right]_{t_{lw}}^{t} \tag{C.18}$$

Finally,

$$\int_{t_{lw}}^{t} dw = [w(t)]_{t_{lw}}^{t} = \frac{1}{-\frac{1}{\tau_c}-\frac{1}{\tau_m}}\, c(t_{lc})m(t_{lm}) \left[e^{-\left(\frac{t-t_{lc}}{\tau_c}\right)-\left(\frac{t-t_{lm}}{\tau_m}\right)}\right]_{t_{lw}}^{t} \tag{C.19}$$

Since weights are evaluated when a spike arrives to a post-synaptic core, we need two versions of the weight update. If the last weight update was done when a standard spike arrived ($t_{lw} = t_{lc}$)

$$\Delta w = \frac{c(t_{lc})m(t_m)}{-\frac{1}{\tau_c}-\frac{1}{\tau_m}} \left[e^{-\left(\frac{t-t_{lc}}{\tau_c}\right)} e^{-\left(\frac{t-t_{lm}}{\tau_m}\right)} - e^{-\left(\frac{t_{lc}-t_{lm}}{\tau_m}\right)}\right] . \tag{C.20}$$

Similarly, if the last weight update was done on a dopamine spike arrival ($t_{lw} = t_{lm}$),

$$\Delta w = \frac{c(t_{lc})m(t_{lm})}{-\frac{1}{\tau_c}-\frac{1}{\tau_m}} \left[e^{-\left(\frac{t-t_{lc}}{\tau_c}\right)} e^{-\left(\frac{t-t_{lm}}{\tau_m}\right)} - e^{-\left(\frac{t_{lm}-t_{lc}}{\tau_c}\right)}\right] . \tag{C.21}$$

Appendix D

Software repositories

The developed software can be found at the following locations:

Image conversion dataset

`https://github.com/NEvision/NE15`

NVS emulator (PyDVS)

`https://github.com/chanokin/pyDVS`

Vision modules

`https://github.com/chanokin/vision`

Voltage-change plasticity

`https://github.com/SpiNNakerManchester/sPyNNaker/tree/dvdt_plasticity`
`https://github.com/SpiNNakerManchester/sPyNNaker7/tree/dvdt_plasticity`