Chapter 7

Learning in Neural Networks

By Petruț Bogdan, Garibaldi Pineda García, Michael Hopkins, Edward Jones, James Knight and Adam Perrett

DOI: 10.1561/9781680836530.ch7

Published in *SpiNNaker – A Spiking Neural Network Architecture* by S. Furber and P. Bogdan (eds.). 2020. ISBN 978-1-68083-596-0. E-ISBN 978-1-68083-653-0.

Suggested citation: Petruț Bogdan *et al.* 2020. "Learning in Neural Networks" in *SpiNNaker – A Spiking Neural Network Architecture*. Edited by S. Furber and P. Bogdan. pp. 209–266. Now Publishers. DOI: 10.1561/9781680836530.ch7.

now
the essence of knowledge

If you don't sleep the very first night after learning, you lose the chance to consolidate those memories, even if you get lots of 'catch-up' sleep thereafter. In terms of memory, then, sleep is not like the bank. You cannot accumulate a debt and hope to pay it off at a later point in time. Sleep for memory consolidation is an all-or-nothing event.

— Matthew Walker

A very important set of open questions in Neuroscience are related to learning, from how addiction rewires our brains to how you can remember where you parked your car or left your bike this morning, but can't remember why you entered the kitchen. Neural memories, whether artificial or biological, seem to operate over multiple time scales. Very short-term memories are fast, but limited so get overwritten often. Long-term memories can stick around a lifetime, but they take a good night's sleep to consolidate. Whether through sleep spindles or one-shot learning, brains utilise synaptic plasticity to store these patterns of activity that we call memories, concepts or motor actions.

This chapter is concerned with the motivation, design and implementation behind mimicking biological learning rules with a focus on, you guessed it, SpiNNaker. It starts by presenting Spike-timing-dependent plasticity (STDP) operating in an unsupervised fashion based on relative spike times of the pre- and post-synaptic neurons or based on the sub-threshold membrane potential. This is

followed by a model of STDP modulated by the presence of an additional signal and operating on eligibility traces. Longer-term mechanisms in the form of structural plasticity, involving the rewiring of connections between the neurons, and (very long-term) neuroevolution close out the chapter.

7.1 Sizing Up the (Biological) Competition

A quick detour: Let's talk brains.

The combination of the size of the human brain, in terms of the number of neurons ($\approx 8.8 \times 10^{10}$ [7]) and synapses ($\approx 1.5 \times 10^{14}$ [180, 187]) and the different possible scales of exploration, from gene expression to behaviour, makes the task of mapping the human brain intractable. As a result, our approach focuses on the simulation of brain regions, relying on sparse, but strategic data. Previous attempts have usually been data driven, attempting to replicate at some level the operation of brain cells [100], regions [154] or processes [29, 118], or concept driven, wherein the modellers are interested in harnessing the computational power of whatever it is they are modelling [11, 54, 124]. Further, while the brain is currently the gold standard for computational power, function and flexibility, and a lot of effort is being invested in engineering systems with similar performance, it might be worth remembering that the brain is the result of millions of years of evolution. In its current form, it is a tangled mess that various institutions and large-scale projects (e.g. Amunts *et al.* [4]) are trying to mine for strategic data, but it is not all gold mine.

Some 'features' of the brain are short-cuts, heuristics to deal with vast amounts of sensory information, and so they are fallible [28].

Designing neural circuits that could overcome biological limitations might be more important than being content with replicating the brain's capabilities. Of course, since some computations are not tractable even under the assumption of clairvoyance (e.g. see the scheduling chapter in Christian and Griffiths [36]), heuristics need to be employed and thus accuracy sacrificed.

Regardless of whether we beat brains in terms of performance (spoiler: we haven't yet), the rest of the chapter should be an interesting exploration of learning techniques in SNNs.

7.2 Spike-Timing-Dependent Plasticity

In this section, we will consider only the changing of the strength of *existing* connections and, in this context, Hebb's postulate indicates that connections between neurons which persistently fire at the same time will be strengthened. Neurons

which persistently fire at the same time are likely to do so because they respond to similar or related stimuli.

Bliss and Lømo [20] provided the first evidence to support this hypothesis by measuring how – if two connected neurons are stimulated simultaneously – the synaptic connections between them are strengthened. In networks of rate-based neurons, this behaviour has been modelled using rules such as the Bienenstock, Cooper, Munroe (BCM) rule [18] and Oja's rule [184]. However, the focus of this section is on SNNs, and in such networks, the timings of individual spikes have been shown to encode both temporal and spatial information. Therefore, in this section, we focus on STDP – a form of synaptic plasticity capable of learning such timings.

In Section 7.2.1, we outline some of the experimental evidence supporting STDP and discuss how STDP can be modelled in networks of spiking neurons. Then, in Section 7.2.2, we discuss how STDP has previously been implemented on SpiNNaker and other distributed systems.

We have developed a new SpiNNaker STDP implementation which has both lower algorithmic complexity than prior approaches and employs new low-level optimisations to exploit the ARM instruction set better. Improving the performance of previous SpiNNaker STDP implementations is an important aspect of this work. It is analysed and presented in detail by Knight [129]. Finally, in Section 7.2.3, we discuss this implementation in depth. This new implementation is now a key component of the SpiNNaker software developed as part of the HBP which aims to provide a common platform for running PyNN simulations on SpiNNaker, BrainScaleS and HPC platforms.

7.2.1 Experimental Evidence for Spike-Timing-Dependent Plasticity

Levy and Steward [142] showed that if the experiment performed by Bliss and Lømo [20] was repeated with a delay between the stimulation of two neurons, then the magnitude of the increase in weight could be reduced or even reversed. Subsequently, Bi and Poo [16] measured the changes in synaptic efficacy induced in the synapses of hippocampal neurons by pairs of pre- and post-synaptic spikes with different relative timings. The relationship between the magnitude of these changes and the relative timing of the pre- and post-synaptic spikes is known as STDP, and the data recorded by Bi and Poo suggest that it reinforces causality between the firing of the pre- and post-synaptic neurons. When a pre-synaptic spike arrives before a post-synaptic spike is emitted, the synapse is *potentiated* (strengthened). However, if a pre-synaptic spike arrives after a post-synaptic spike has been emitted, the synapse is *depressed* (weakened). Furthermore, the data recorded by Bi and Poo

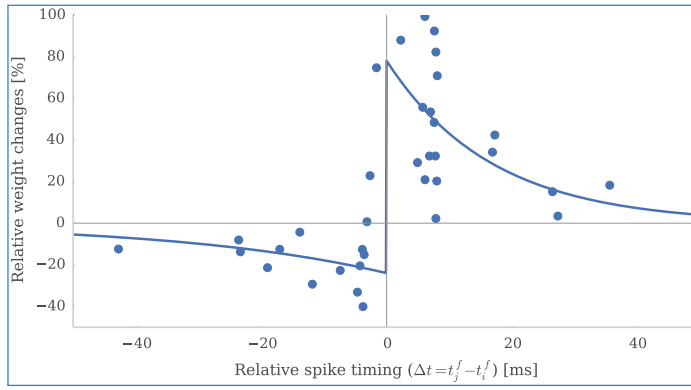

Figure 7.1. Excitatory STDP curve. Each dot represents the relative change in synaptic efficacy after 60 pairs of spikes. After Bi and Poo [16].

suggest that the magnitude of changes in synaptic efficacy (Δw_{ij}) is related to the relative spike timings with the following exponential functions (Figure 7.1):

$$\Delta w_{ij} = \begin{cases} F_+(w_{ij}) \exp\left(-\frac{\Delta t}{\tau_+}\right) & \text{if } \Delta t > 0 \\ F_-(w_{ij}) \exp\left(\frac{\Delta t}{\tau_-}\right) & \text{if } \Delta t \leq 0 \end{cases} \tag{7.1}$$

Where $\Delta t = t_j - t_i$ represents the relative timing of pre- and post-synaptic spikes, $\tau_\pm$ defines the time constant of the exponentials and the $F_\pm$ functions define how the magnitude of the change in weight depends on the current synaptic efficacy.

Bi and Poo [16] measured how Δw_{ij} depended on the previous value of w_{ij}. In Figure 7.2, we redraw their data on a double-logarithmic scale and fit straight lines to the potentiation and depression components (as suggested by Morrison *et al.* [170]). The nature of the $F_\pm$ functions is indicated by the gradient of each line. Since the trend line through the depression data has a gradient of -1, it would suggest that F_- is linearly proportional to the weight. However the nature of F_+ is less clear as the trend line through the potentiation data has a gradient of 0.4. Gütig *et al.* [86] and Morrison *et al.* [170] proposed using the power law function $F_+ \propto {w_{ij}}^\mu$ to represent the magnitude of the change in weight in response to potentiation; $\mu = 0$ makes the weight update independent of the previous weight; and $\mu = 1$ makes the update linearly proportional to the previous weight. With $\mu = 0.4$, the linear fit to the potentiation data recorded by Bi and Poo can be obtained.

Another common means of describing STDP rules is by using 'trace variables' [170, 171, 210] which get updated when pre- and post-synaptic spikes occur and represent the combined effects of the preceding pre- and post-synaptic spikes. For example, the interactions between individual pairs of spikes described by

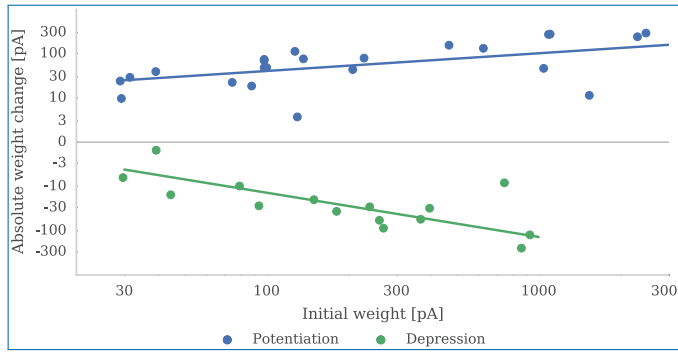

Figure 7.2. Absolute change in synaptic efficacy after 60 spike pairs. Potentiation is induced by spike pairs where the pre-synaptic spike precedes the post-synaptic spike by 2.3 ms to 8.3 ms. Depression is induced by spike pairs in which the post-synaptic spike precedes the pre-synaptic spike by 3.4 ms to 23.6 ms. The upper blue line is a linear fit to the potentiation data with slope: 0.4. The lower green line is a linear fit to the depression data with slope: −1. After Morrison *et al.* [170].

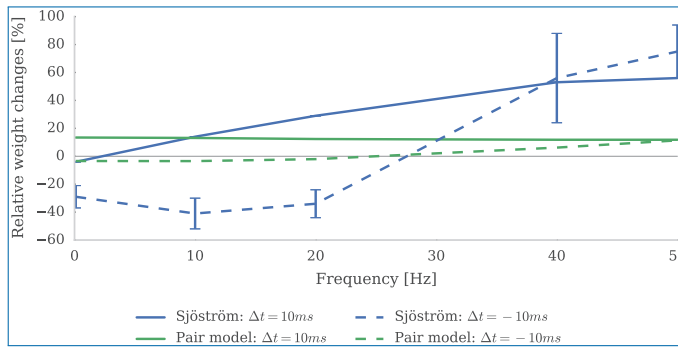

Figure 7.3. Fitting a pair-based STDP model with $\tau_+ = 16.8$ ms and $\tau_- = 33.7$ ms to data from Sjöström *et al.* [229] by minimising the mean squared error fails to reproduce frequency effects. Blue lines and data points redrawn from Sjöström *et al.* and the green lines show the best fit obtained by the pair-based STDP model. After Pfister [192].

Equation 7.1 can alternatively be modelled based on pre- (s_i) and post-synaptic (s_j) trace variables:

$$\frac{ds_i}{dt} = -\frac{s_i}{\tau_+} + \sum_{t_i^f} \delta(t - t_i^f) \tag{7.2}$$

$$\frac{ds_j}{dt} = -\frac{s_j}{\tau_-} + \sum_{t_j^f} \delta(t - t_j^f) \tag{7.3}$$

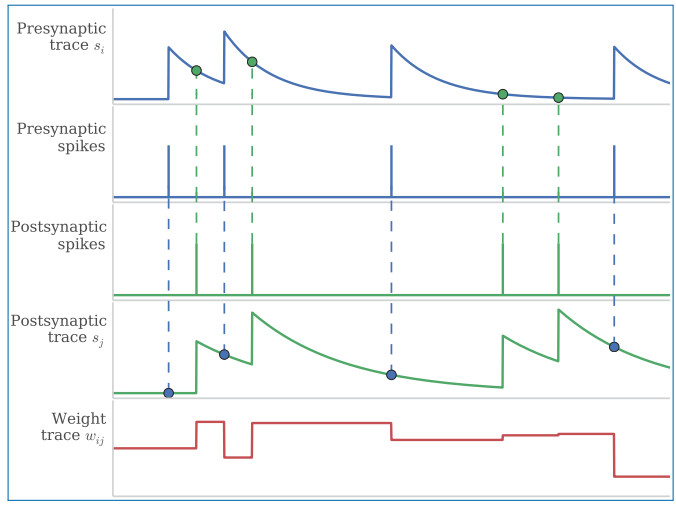

Figure 7.4. Calculation of weight updates using pair-based STDP traces. Pre- and post-synaptic traces reflect the activity of pre- and post-synaptic spike trains. Potentiation is calculated at each post-synaptic spike time by sampling the pre-synaptic trace (green circle) to obtain a measure of recent pre-synaptic activity. Depression is calculated at each pre-synaptic spike time by sampling the post-synaptic trace (blue circle) to obtain a measure of recent post-synaptic activity. Weight dependence is additive. After Morrison *et al.* [171].

Pre- and post-synaptic spikes occurring at t_i^f and t_j^f, respectively, are modelled using Dirac delta functions (δ) and, as the top 4 panels of Figure 7.4 show, the trace variables represent a low-pass filtered version of these spikes. These dynamics can be thought of as representing chemical processes. For example s_i can be viewed as a model of glutamate neurotransmitters which, having crossed the synaptic cleft from the pre-synaptic neuron, bind to receptors on the post-synaptic neuron and are reabsorbed with a time constant of τ_+. Building on this work, Section 7.4 presents an implementation of neuromodulated STDP simulated on SpiNNaker.

As the dashed blue lines in Figure 7.4 illustrate, when a pre-synaptic spike occurs at time t_i^f, the s_j trace can be sampled to obtain the combined depression caused by the pairs made between this pre-synaptic spike and all preceding post-synaptic spikes. Similarly, as the dashed green lines in Figure 7.4 illustrate, when a post-synaptic spike occurs at time t_j^f, the s_i trace can be sampled, leading to the following equations for calculating depression (Δw_{ij}^-) and potentiation (Δw_{ij}^+):

$$\Delta w_{ij}^-(t_i^f) = F_-(w_{ij}) s_j(t_i^f) \tag{7.4}$$

$$\Delta w_{ij}^+(t_j^f) = F_+(w_{ij}) s_i(t_j^f) \tag{7.5}$$

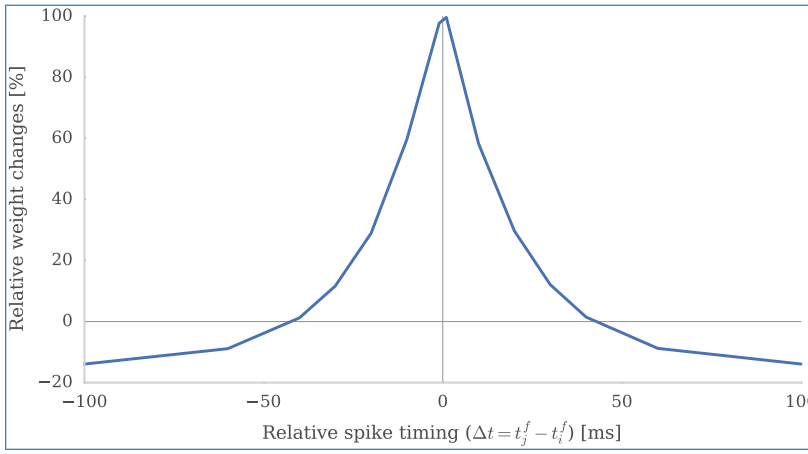

Figure 7.5. Inhibitory STDP curve. The relative change in synaptic efficacy after 60 pairs of spikes. After Vogels *et al.* [261].

Bi and Poo [16] recorded the data plotted in Figures 7.1 and 7.2 from rat hippocampal neurons, but subsequent studies have revealed similar relationships – albeit with different time constants and polarities – in other brain areas [210]. Specifically, in the neocortex, excitatory synapses appear to exhibit STDP with similar asymmetrical kernels to hippocampal neurons, whereas inhibitory synapses have a symmetrical kernel similar to that shown in Figure 7.5.

While rules that consider pairs of spikes provide a good fit for the data measured by Bi and Poo, they cannot account for effects seen in more recent experimental data. Sjöström *et al.* [229] stimulated cortical neurons with pairs of pre- and post-synaptic spikes separated by a constant 10 ms but with between 20 ms and 10 s separating the pairs. When the time between the pairs approaches the time constants defining the temporal range of the pair-based STDP rule, spikes from neighbouring pairs begin to interact. As shown in Figure 7.3, this interaction then cancels out the potentiation or depression that the original pair should have elicited.

Several extensions to the STDP rule have been proposed which take into account the effect of multiple preceding spikes including the 'triplet rule' proposed by Pfister [192]. In this rule, the effect of earlier spikes is modelled using a second set of traces (s_i^2 and s_j^2) with longer time constants τ_x and τ_y:

$$\frac{ds_i^2}{dt} = -\frac{s_i^2}{\tau_x} + \sum_{t_i^f} \delta(t - t_i^f) \tag{7.6}$$

$$\frac{ds_j^2}{dt} = -\frac{s_j^2}{\tau_y} + \sum_{t_j^f} \delta(t - t_j^f) \tag{7.7}$$

To incorporate the effect of these traces into the weight updates, Pfister also extended Equations 7.4 and 7.5:

$$\Delta w_{ij}^{-}(t_i^f) = s_j(t_i^f)(A_2^- + A_3^- s_i^2(t_i^f - \epsilon)) \tag{7.8}$$

$$\Delta w_{ij}^{+}(t_j^f) = s_i(t_j^f)(A_2^+ + A_3^+ s_j^2(t_j^f - \epsilon)) \tag{7.9}$$

Where ϵ is a small positive constant used to ensure that the second set of s^2 traces is sampled just *before* the spike occurs at t_i^f or t_j^f. This rule has an explicitly additive weight dependence with the relative effect of the four traces controlled by the four free parameters A_2^+, A_2^-, A_3^+ and A_3^-. Pfister fitted these free parameters to the data obtained by Sjöström *et al.* [229] and, as shown in Figure 7.6, demonstrated that the rule can accurately reproduce the frequency effect measured by Sjöström *et al.*

The trace-based models we have discussed so far assume that all preceding spikes can affect the magnitude of STDP weight updates. However experimental data [229] suggest that this might not be the case and that basing pair-based STDP weight updates on only the most recent spike can improve the fit of these models to experimental data. This 'nearest-neighbour' spike interaction scheme can be implemented in a trace-based model by resetting the appropriate trace to 1 when a spike occurs rather than by incrementing it by 1. Pfister also investigated the effect of different spike interaction schemes on their triplet rule but found it had

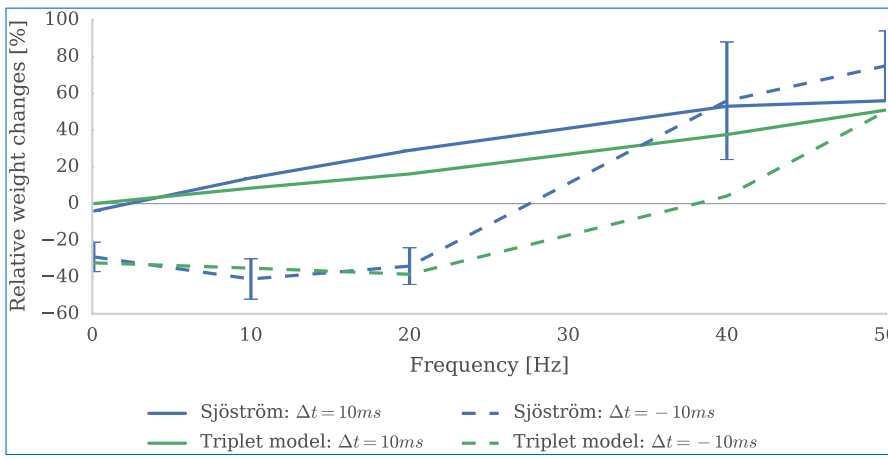

Figure 7.6. Fitting triplet STDP model with $\tau_+ = 16.8$ ms, $\tau_- = 33.7$ ms, $\tau_x = 101$ ms and $\tau_y = 125$ ms to the data recorded by Sjöström *et al.* [229] by minimising mean squared error effectively reproduces frequency effects. Blue lines and data points (with errors) redrawn from Sjöström *et al.* and the green lines show the best fit obtained by the triplet STDP model. After Pfister [192].

no significant effect on its fit to the data recorded by Sjöström *et al.* This suggests that alternative spike-pairing schemes may simply be another means of overcoming some of the limitations of pair-based STDP models.

7.2.2 Related Work

Implementing the STDP rules discussed in Section 7.2.1 in a naïve manner is relatively trivial. However, implementing them in a manner suitable for large scale simulation on a distributed system such as SpiNNaker is more difficult.

The efficient access to synaptic weights required by the event-driven synaptic processing algorithm is facilitated by storing the synaptic matrices in a row-major format. Consequently, when a pre-synaptic spike arrives, weight updates (Equation 7.4) can be evaluated on a row which is *contiguous* in memory. However, when a post-synaptic spike is emitted, weight updates (Equation 7.3) must be evaluated on a *non-contiguous* synaptic matrix *column*. Accessing synaptic matrix columns is problematic at the hardware level as the SpiNNaker DMA controller can fetch only contiguous blocks of data. Moreover, because connectivity in the neocortex is relatively sparse, synaptic matrices are represented using a compressed sparse row structure which does *not* provide efficient access to matrix columns. To remove the need for column accesses, all of the STDP implementations presented in this section defer outgoing post-synaptic spikes to allow post-synaptic weight updates to be deferred until the next *pre-synaptic* spike occurs.

An additional problem regards synaptic delays. Delays are simulated on SpiNNaker by inserting synaptic weights into an input ring buffer. However the relative timing of pre- and post-synaptic spikes, and therefore the outcome of STDP, depends on how much of this total delay occurs in the pre-synaptic axon and how much in the post-synaptic dendritic tree. As shown in Figure 7.7, pre-synaptic spikes travelling down the pre-synaptic axon to the synapse incur an 'axonal delay' and post-synaptic spikes propagating back through the post-synaptic dendritic tree to the synapse incur a 'dendritic delay'.

The approaches discussed in this section differ largely in the algorithms and data structures they use to perform the deferral of post-synaptic spikes and to

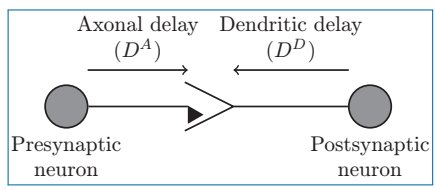

Figure 7.7. The dendritic and axonal components of synaptic delay. After Morrison *et al.* [171].

incorporate synaptic delays into the STDP processing. Jin *et al.* [121] were the first to implement STDP on SpiNNaker. They assumed that the whole synaptic delay was axonal implying that, as pre-synaptic spikes reach the synapse before this axonal delay has been applied, they too must be buffered. Jin *et al.* used a compact data structure for buffering both pre-synaptic and post-synaptic spikes containing the time at which the neuron last spiked and a bit field, the bits of which indicate previous spikes in a fixed window of time. Consequently, only a small amount of DTCM is required to store the deferred spikes associated with each post-synaptic neuron. However, because this approach does not use the trace-based STDP model (Section 7.2.1), the effect of *all possible pairs* of pre- and post-synaptic spikes must be calculated separately using Equation 7.1. Additionally, the bit field based recording of history – while compact – represents only a fixed window of time meaning that only a very small number of spikes from slow firing neurons can ever be processed.

Diehl and Cook [50] developed the first trace-based STDP implementation for SpiNNaker. To store the pre- and post-synaptic traces, they extended each synapse in the synaptic row to contain the values of the traces at the time of the last update. They allowed synapses to have arbitrary axonal and dendritic delays meaning that, like Jin *et al.*, they stored a history of both pre- and post-synaptic spikes. However, rather than using a bit field to store this, they used a fixed-size circular buffer to store the spike times. This data structure is not only faster to iterate over than a bit field but also holds a constant number of spikes, regardless of the firing rates of the pre- and post-synaptic neurons. However, these buffers can still overflow, leading to spikes not being processed if the pre- and post-synaptic firing rates are too different. For example, consider a buffer with space for ten entries being used to defer the spikes from a post-synaptic neuron firing at 10 Hz. If one of the neuron's input synapses only receives spikes (and is thus updated) at 0.1 Hz, there is insufficient buffer space for all $100 = \frac{10\,\text{Hz}}{0.1\,\text{Hz}}$ of the post-synaptic spikes that occur between the updates. Using these spike histories, Diehl and Cook developed an algorithm to perform trace-based STDP updates whenever the synaptic matrix row associated with an incoming spike packet is retrieved from the SDRAM. The algorithm loops through these synapses and, for each one, iterates through the buffered pre- and post-synaptic spikes in the order that they occurred since the last update (taking into account the dendritic and axonal delays). The effect of each buffered spike is then applied to the synaptic weight (using Equation 7.4 for pre-synaptic spikes and Equation 7.5 for post-synaptic spike) and the appropriate trace updated (using Equation 7.2 for pre-synaptic spikes and Equation 7.3 for post-synaptic spike). Diehl and Cook measured the performance of their approach using a benchmark network of 50 LIF neurons stimulated by a large number of 250 Hz Poisson spike sources connected with 20% sparsity. Using this network, they showed that their

approach could process 500×10^3 incoming synaptic events per second compared to the 50×10^3 achievable using the approach developed by Jin *et al.*

There are many similarities between simulating large spiking neural networks on SpiNNaker and on other distributed computer systems – including the two problems identified at the beginning of this section. In the distributed computing space, Morrison *et al.* [170] addressed these in ways highly relevant to a SpiNNaker implementation. Although the nodes of the distributed systems they targeted do not have to access synaptic matrix rows using a DMA controller, accessing non-contiguous memory is also costly on architectures with hardware caches. Therefore, post-synaptic weight updates still need to be deferred until a pre-synaptic spike. As each node has significantly more memory, Morrison *et al.* use a dynamic data structure to guarantee that all deferred post-synaptic spikes get processed.

Morrison *et al.* simplify the model of synaptic delay by supporting only configurations where the axonal delay is shorter than the dendritic delay. This simplification allows pre-synaptic spikes to be processed immediately as it guarantees that post-synaptic spikes emitted before the axonal delay has elapsed will never 'overtake', and thus need to be processed before, the pre-synaptic spike.

This simplification means that only the time of the last pre-synaptic spike and the value of the pre-synaptic trace at that time need to be stored with each synaptic matrix row. Based on this simplification, the algorithm developed by Morrison *et al.* loops through each synapse in the row and, for each one, loops through the buffered post-synaptic spikes. The effect of each buffered spike is then applied to the synaptic weight (using Equation 7.5). After all of the post-synaptic spikes have been processed, the effect of the pre-synaptic spike that instigated the update is applied to the synaptic weight (using Equation 7.4). Once all of the synapses in the row have been processed, the pre-synaptic trace is updated (using Equation 7.2).

To assess the relative algorithmic complexity of the approaches presented in this section, we can consider the situation where an STDP synapse is updated based on N^{pre} pre-synaptic and N^{post} post-synaptic spikes. In the approach developed by Jin *et al.* [121], each pair of spikes is processed individually and the complexity is $O(N^{pre}N^{post})$. However, by using a trace-based approach, Diehl and Cook [50] reduced this complexity to $O(N^{pre} + N^{post})$ and Morrison *et al.* [170] further reduced this to $O(N^{post})$ by removing the need to buffer pre-synaptic spikes.

7.2.3 Implementation

The best performing SpiNNaker STDP implementation presented in the previous section was that developed by Diehl and Cook [50]. Their benchmark indicated that, using their implementation, a SpiNNaker core could process up to 500×10^3 incoming synaptic events per second, compared with 5×10^6 events

for non-plastic synapses. As this corresponds to a significant reduction in the size of model a given SpiNNaker machine can simulate, improving the performance of STDP is an important part of enabling large-scale neocortical simulation on SpiNNaker.

We have developed a new SpiNNaker STDP implementation based on the algorithm developed by Morrison *et al.* [170] which has lower algorithmic complexity than previous SpiNNaker implementations and employs new low-level optimisations to exploit the ARM instruction set better. In this section, we present the details of this new implementation and demonstrate how it addresses the previously identified problems with distributed simulation of STDP.

As discussed in Section 7.2.2, Diehl and Cook used a fixed-sized data structure with space for 10 events to store the post-synaptic history. Using this system, if more than 10 post-synaptic spikes backpropagate to a synapse between updates, some will be lost. Based on the distributions of cortical neuron firing rates in rats and macaque monkeys presented by Buzsáki and Mizuseki [30], we can derive the distribution of firing rate ratios between pairs of neurons shown in Figure 7.8. Based on these ratio distributions, we can determine that a buffer with 10 entries will be sufficient to handle the activity at 90% of synapses. However to prevent post-synaptic spikes being lost when the pre- and post-synaptic neurons have very different firing rates, we developed an additional mechanism called 'flushing' to force the processing of these spikes. This mechanism uses one bit in the 32-bit ID associated with each neuron to signify whether the neuron is emitting a 'flush' or an actual spike event. To determine when these events should be sent, each neuron tracks its Inter-spike interval (ISI) and, if this is *bufferSize* times longer than the ISI corresponding to the maximum firing rate of the network, a flush event is emitted.

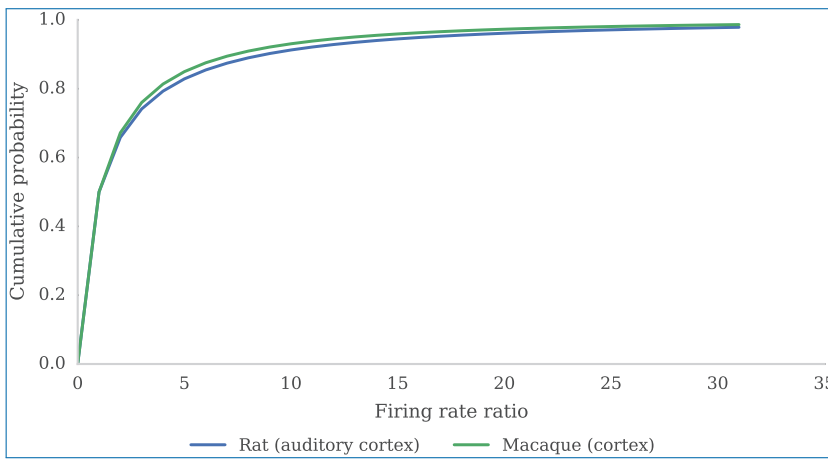

Figure 7.8. Ratio distributions of cortical firing rates. Calculated from firing rate distributions presented by Buzsáki and Mizuseki [30].

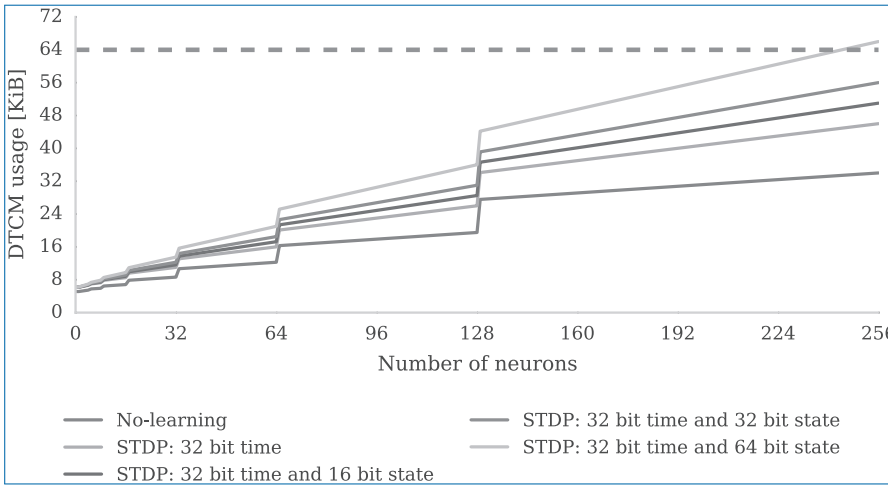

Figure 7.9. DTCM memory usage of STDP event storage schemes. The memory usage of other components is based on the current SpiNNaker tools. All trace-based schemes assume times are stored in a 32-bit format and traces in a 16-bit format, with two look-up tables with 256 16-bit entries providing exponential decay. The dashed horizontal line shows the maximum available DTCM.

Figure 7.9 shows the local memory requirements of post-synaptic history structures with capacity for 10 entries of different sizes. To implement STDP rules such as the triplet rule discussed in Section 7.2.1, each entry needs to be large enough to hold not only a spike time but also two trace values. Figure 7.9 suggests that, to avoid further reductions in the number of neurons that each SpiNNaker core can simulate, each of these traces should be represented as a 16-bit value. Using 16-bit trace entries has an additional advantage as the ARM 968 CPU used by SpiNNaker includes single-cycle instructions for multiply and multiply-accumulate operations on signed 16-bit integers [62]. These instructions allow additive weight updates such as $w_{ij} \leftarrow w_{ij} + s_j \exp\left(\frac{-dt}{tau}\right)$ to be performed using a single *SMLAxy* instruction and, when implementing rules such as the triplet rule that require two traces, they provide an efficient means of operating on pairs of 16-bit traces stored within a 32-bit field.

The range of fixed-point numeric representations is static. Thus, the optimal representation for storing traces must be chosen ahead of time based on the maximum expected value. We can calculate this by considering the value of a trace x with time constant τ after n spikes emitted at f Hz:

$$x(n) = \sum_{i=0}^{n} e^{-\frac{i}{\tau f}} \tag{7.10}$$

This can be rearranged into the form of a geometric sum:

$$x(n) = \sum_{i=0}^{n} (e^{-\frac{1}{\tau f_{max}}})^i \tag{7.11}$$

Which has the value:

$$x(n) = \frac{1 - (e^{-\frac{1}{\tau f}})^n}{1 - e^{-\frac{1}{\tau f}}} \tag{7.12}$$

Since $|e^{-\frac{1}{\tau f}}| < 1$, as $n \to \infty$ this converges to:

$$x_{max} = \frac{1}{1 - e^{-\frac{1}{\tau f}}} \tag{7.13}$$

The sustained firing rate of most neurons is constrained by the time that ion pumps take to return the neuron's membrane potential to its resting potential. This generally limits a neuron's maximum firing rate to around 100 Hz but, as Gittis *et al.* [79] discuss, there are mechanisms that can overcome this limit. For example, vestibular nucleus neurons can maintain sustained firing rates of around 300 Hz. Figure 7.10 shows that – based on this worst-case maximum firing rate – 4 integer bits are required to store traces with time constants in the range fitted to the data recorded by Bi and Poo [16]. Therefore, a 16-bit fixed-point numeric representation with 4 integer, 11 fractional bits and a sign bit is the optimal choice for representing the traces required for pair-based STDP.

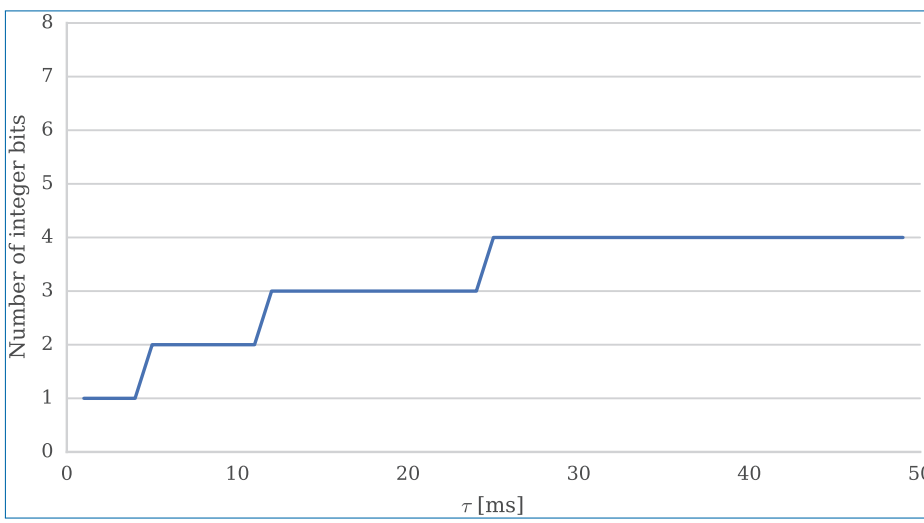

Figure 7.10. Number of integer bits required to represent traces of a 300 Hz spike train with different time constants.

In the PyNN programming interface, STDP learning rules are defined in terms of three components:

The timing dependence: Defines how the relative timing of the pre- and post-synaptic spikes affects the magnitude of the weight update.

The weight dependence: Defines how the current synaptic weight affects the magnitude of the weight update (the F_+ and F_- functions discussed in Section 7.2.1).

The voltage dependence: Defines how the membrane voltage of the post-synaptic neuron affects the magnitude of the weight update.

Adding a voltage dependence to the type of event-based STDP implementation discussed here presents several challenges beyond the scope of this section. However, one such voltage-dependent implementation is described in Section 7.3.

In this section, we implement only the timing and weight dependencies supported by PyNN. So as to allow users of the HBP software not only to select from the weight dependencies specified by PyNN but also to implement their own easily, this implementation defines simple interfaces which timing and weight dependencies must implement. Timing dependencies must define the correct types for the pre- and post-synaptic states (s_i and s_j, respectively), functions to update pre- and post-synaptic trace entries based on the time of a new spikes (*updatePreTrace* and *updatePostTrace*, respectively) and functions to apply the effect of deferred pre- and post-synaptic spikes to a synaptic weight (*applyPreSpike* and *applyPostSpike*, respectively). Algorithm 1 shows an implementation of the functions required to implement pair-based STDP using this interface. The *updatePreTrace* adds the effect of a new pre-synaptic spike at time t to the pre-synaptic trace by decaying the value of s_i calculated at the time of the last spike ($t^{\text{lastSpike}}$) and adding 1 to represent the effect of the new spike (the closed-form solution to Equation 7.2 between two t_i^f s). Similarly, the *applyPreSpike* function samples the post-synaptic trace by decaying the value of s_j calculated at the time of the last post-synaptic spike (t_j) (the $s_j(t_i^f)$ term of Equation 7.4).

To decouple the timing and weight dependencies, the *applyPreSpike* and *applyPostSpike* functions in the timing dependence call the *applyDepression* and *applyPotentiation* functions provided by the weight dependence rather than directly manipulating w_{ij} themselves. Algorithm 2 shows an implementation of *applyDepression* which performs an additive weight update.

Algorithm 3 is the result of combining the simplified delay model proposed by Morrison *et al.* [170] with the flushing mechanism and the interfaces for timing and weight dependencies discussed in this section. The algorithm begins by looping through each post-synaptic neuron (j) in the row and retrieving a list of the

Algorithm 1 Pair-based STDP timing-dependence implementation. Equivalent *updatePostTrace* and *applyPostTrace* functions are omitted for brevity.

function UPDATEPRETRACE(s_i, t, $t^{\text{lastSpike}}$)
  $\Delta t \leftarrow t - t^{\text{lastSpike}}$ **return** $s_i \cdot \exp\left(-\frac{\Delta t}{tau}\right) + 1$

function APPLYPRESPIKE(w_{ij}, t, t_j, s_j)
  $\Delta t \leftarrow t - t_j$
  if $\Delta t \neq 0$ **then return** *applyDepression* $\left(w_{ij}, s_j \cdot \exp\left(-\frac{\Delta t}{tau}\right)\right)$
  else return w_{ij}

Algorithm 2 Additive weight-dependence implementation. Equivalent *applyPotentiation* function is omitted for brevity.

function APPLYDEPRESSION(w_{ij}, d) **return** $w_{ij} + A^+ \cdot d$

Algorithm 3 The new SpiNNaker STDP algorithm

function PROCESSROW(t, $flush$, $t^{\text{lastUpdate}}$, $t^{\text{lastSpike}}$, s_i, $synapses$)
  for all (j, w_{ij}, d^A, d^D) **in** $synapses$ **do**
    $history \leftarrow getHistoryEntries(j, t^{\text{lastUpdate}} + d^A - d^D, t + d^A - d^D)$

    for all (t_j, s_j) **in** $history$ **do**
      $w_{ij} \leftarrow applyPostSpike(w_{ij}, t_j + d^D, t^{\text{lastSpike}} + d^A, s_i)$

    if not $flush$ **then**
      $(t_j, s_j) \leftarrow getLastHistoryEntry(t + d^A - d^D)$
      $w_{ij} \leftarrow applyPreSpike(w_{ij}, t + d^A, t_j + d^D, s_j)$
      $addWeightToRingBuffer(w_{ij}, j)$

  if not $flush$ **then**
    $s_i \leftarrow updatePreTrace(s_i, t, t^{\text{lastSpike}})$
    $t^{\text{lastSpike}} \leftarrow t$
  $t^{\text{lastUpdate}} \leftarrow t$

times (t_j) at which that neuron spiked between $t^{\text{lastUpdate}}$ and t and its state at that time (s_j) (taking into account the dendritic (D^D) and axonal (D^A) delays associated with each synapse). The algorithm continues by looping through each post-synaptic spike and calling the *applyPostSpike* function to apply the effect of the interaction between the post-synaptic spike and the pre-synaptic spike that occurred at $t^{\text{lastSpike}}$ to the synapse. If the update was instigated by a pre-synaptic spike rather than a flush, the *applyPreSpike* function is called to apply the effect of the interaction between the pre-synaptic spike and the most recent post-synaptic spike to the

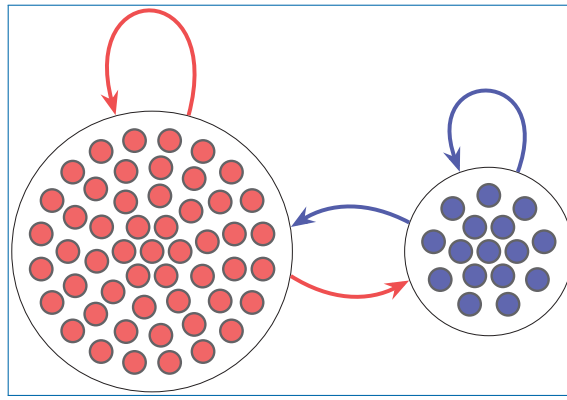

Figure 7.11. A random balanced network consisting of recurrently and reciprocally connected populations of excitatory neurons (red filled circles) and inhibitory neurons (blue filled circles). Excitatory connections are illustrated with red arrows and inhibitory connections with blue arrows.

synapse. Once all events are processed, the fully updated weight is added to the input ring buffer. If the update was instigated by a pre-synaptic spike rather than a flush, after all the synapses are processed, the pre-synaptic state stored in the header of the row (s_i) is updated by calling the *updatePreTrace* function and $t^{\text{lastSpike}}$ and $t^{\text{lastUpdate}}$ are set to the current time. If, however, the update was instigated by a flush event, only $t^{\text{lastUpdate}}$ is updated to the current time, meaning that the interactions between future post-synaptic events and the last pre-synaptic spike will continue to be calculated correctly.

7.2.4 Inhibitory Plasticity in Cortical Networks

One of the simplest models of a cortical network consists of a population of excitatory neurons and a smaller population of inhibitory neurons, sparsely connected with the recurrent and reciprocal synapses shown in Figure 7.11, that is, the random balanced network introduced in Chapter 4. Brunel [27] identified that these networks can operate in several well-defined regimes depending on the relative weights of the inhibitory and excitatory synapses. The asynchronous irregular regime has proved of particular interest as it matches firing rate statistics recorded in the neocortex [232] and responds rapidly to small changes in input making it an ideal substrate for computation [262]. However, it is unclear how the carefully balanced synaptic weights required to establish this regime are maintained in the brain. Vogels *et al.* [261] demonstrated that the asynchronous irregular regime can be

Algorithm 4 Inhibitory plasticity timing-dependence implementation. *update PreTrace* and *updatePostTrace* functions are identical to those used by standard STDP and are therefore omitted for brevity.

function APPLYPRESPIKE(w_{ij}, t, t_j, s_j)
$\Delta t \leftarrow t - t_j$ **return** *applyPotentiation* $\left(w_{ij}, s_j \cdot \exp\left(-\frac{\Delta t}{tau}\right) - \alpha\right)$
function APPLYPOSTSPIKE(w_{ij}, t, t_i, s_i)
$\Delta t \leftarrow t - t_i$ **return** *applyPotentiation* $\left(w_{ij}, s_i \cdot \exp\left(-\frac{\Delta t}{tau}\right)\right)$

established using an STDP rule with the type of symmetrical kernel shown in Figure 7.5. We implemented this learning rule using the timing dependence functions defined in Algorithm 4 and used it to reproduce the results presented by Vogels *et al.* using a network of 2,000 excitatory and 500 inhibitory neurons with the parameters listed in Table 7.1.

Without inhibitory plasticity, the network remained in the synchronous regime shown in Figure 7.12(a) in which neurons spiked simultaneously at high rates. However, with inhibitory plasticity enabled on the connection between the inhibitory and the excitatory populations, the neural activity quickly stabilised and, as shown in Figure 7.12(b), the network entered an asynchronous irregular regime in which neurons spiked at a much lower rate.

7.2.5 The Effect of Weight Dependencies

In Section 7.2.1, we discussed how the choice of weight dependence affects the fit of STDP models to biological data. Rubin *et al.* [214] showed that different weight dependencies also result in different equilibrium distributions of synaptic weights when neurons with a biologically plausible number of synapses are stimulated with Poisson spike trains. Rubin *et al.* proved that a multiplicative weight dependence results in a unimodal distribution of weights, whereas an additive dependence results in a bimodal distribution.

To demonstrate the flexibility of the SpiNNaker STDP implementation presented here, we reproduced these results empirically using the simple PyNN model described in Table 7.2. The resultant weight distributions are plotted in Figure 7.13. Additive weight dependencies in PyNN specify hard upper and lower bounds and, as Rubin *et al.* predicted, the experiment using the additive weight dependence results in a weight distribution with modes centred at these bounds. Again, as Rubin *et al.* predicted, the experiment using the multiplicative weight dependence results in a unimodal weight distribution.

Table 7.1. Model description of the inhibitory plasticity network. After Nordlie [182]

Model summary		
Populations	Excitatory, inhibitory	
Connectivity	Probabilistic with 2% connection probability	
Neuron model	LIF with exponential current inputs	
Plasticity	Inhibitory plasticity (Vogels *et al.* [261])	
Populations		
Name	Elements	Size
Excitatory	LIF	2,000
Inhibitory	LIF	500
Connectivity		
Source	Target	Weight
Excitatory	Inhibitory	0.03 nA
Excitatory	Excitatory	0.03 nA
Inhibitory	Inhibitory	0.3 nA
Inhibitory	Excitatory	0 nA
Neuron and synapse model		
Type	LIF with exponential current inputs	
Parameters	$g_L = 0.01$ µS leak conductance	
	$C = 0.2$ nF membrane capacitance	
	$V_{thresh} = -50$ mV threshold voltage	
	$V_{reset} = V_{rest} = -60$ mV reset/resting voltage	
	$\tau_{syn}^{exc} = 5$ ms excitatory synaptic time constant	
	$\tau_{syn}^{inh} = 10$ ms inhibitory synaptic time constant	
Plasticity		
Type	Inhibitory plasticity (Vogels *et al.* [261])	
	on inhibitory excitatory synapses	
Parameters	$\rho = 0.12$ µS post-synaptic target firing rate	
	$\tau = 20.0$ ms trace time constant	
	η learning rate	

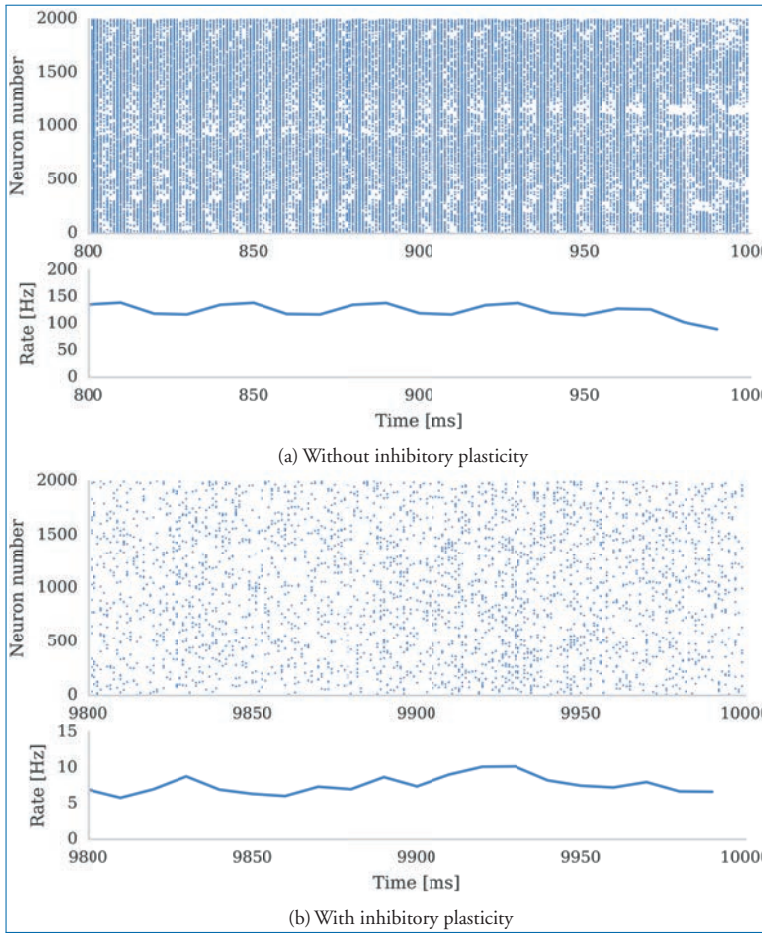

Figure 7.12. The effect of inhibitory plasticity on a random balanced network with 2,000 excitatory and 500 inhibitory neurons. Without inhibitory plasticity, the network is in a synchronous state with all neurons firing regularly at high rates. Inhibitory plasticity establishes the asynchronous irregular state with all neurons firing at approximately 10 Hz.

7.3 Voltage-Dependent Weight Update

Although highly successful, the STDP algorithm has some drawbacks. For example, if the simulator has no memory of pre- and post-synaptic spike times, the algorithm is difficult to implement; furthermore, if the post-synaptic neuron fails to spike, it could be that important information is lost. Bengio *et al.* [14] propose a plasticity rule which is compatible with STDP dynamics and could, in principle, be a way to link machine learning and neuroscience.

Table 7.2. Model description of the synaptic weight distribution network. After Nordlie [182].

Model summary		
Populations	Neurons, stimuli	
Connectivity	All-to-all	
Neuron model	LIF with exponential current inputs	
Plasticity	STDP	
Populations		
Name	Elements	Size
Neurons	LIF	1
Stimuli	Independent 15 Hz Poisson spike trains	1,000
Connectivity		
Source	Target	Weight
Stimuli	Neurons	Uniformly distributed between 0 nA to 0.01 nA
Neuron and synapse model		
Type	LIF with exponential current inputs	
Parameters	$g_L = 0.017$ µS leak conductance	
	$C = 0.17$ nF membrane capacitance	
	$V_{thresh} = -54$ mV threshold voltage	
	$V_{reset} = -60$ mV reset voltage	
	$V_{rest} = -74$ mV resting voltage	
	$\tau_{syn} = 5$ ms synaptic time constant	
Plasticity		
Type	STDP with additive or multiplicative weight dependence	
Parameters	$A^+ = 0.01$ potentiation rate	
	$A^- = 0.0105$ depression rate	
	$\tau_+ = 20.0$ ms trace time constant	
	$\tau_- = 20.0$ ms learning rate	

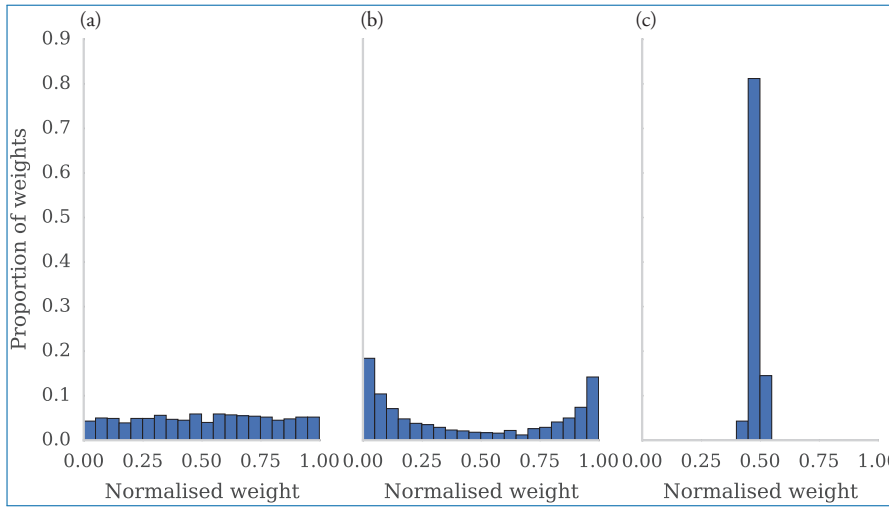

Figure 7.13. Histograms showing: (a) initial uniform distribution of synaptic weights and distribution of synaptic weights following; (b) STDP with additive weight dependence and (c) STDP with multiplicative weight dependence. Simulation consists of a single integrate-and-fire neuron with 1,000 independent 15 Hz Poisson spike sources providing synaptic input.

Table 7.3. Izhikevich neuron model parameters.

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

We implemented this rule using both LIF and Izhikevich neuron models [193]; we only show the results with the latter here. The parameters used in our experiments for the Izhikevich model are shown in Table 7.3. The behaviour of the neuron model when a continuous current is applied is illustrated in Figure 7.14(a).

The blue line depicts the neuron's membrane voltage (v), and the green line shows the behaviour of the auxiliary variable u. Since the membrane voltage is usually noisy, we filter it using the exponential smoothing technique [175]

$$\gamma = e^{-1/\tau_s}, \tag{7.14}$$

$$s(t) = (1-\gamma)v(t) + \gamma s(t-1); \tag{7.15}$$

where τ_s is the temporal constant for the filtering mechanism. The dashed red line is a low-passed version of the membrane voltage (s). The change in synaptic efficacy

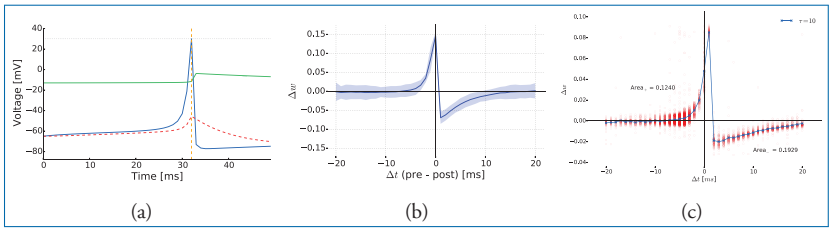

Figure 7.14. Membrane voltage change as a proxy for weight updates. (a) Behaviour of an Izhikevich neuron to a step input; the blue line illustrates the membrane voltage, while the red dashed line shows a low-pass-filtered version of it. (b) shows how an average of weight changes behaves close to STDP when simulated in Python. (c) summarises the average of weight changes simulated on SpiNNaker.

is given by

$$\Delta w = \alpha \times \delta(t - t_{pre}) \times \frac{\Delta s(t)}{\Delta t} \tag{7.16}$$

where $\delta(\cdot)$ is the Kronecker delta function and α is the scaling factor and could be used as the learning rate. To test whether this adjustment to the original learning rule still produces similar results (i.e. STDP-compatible behaviour), we establish an experimental set-up similar to the one presented by Bengio *et al.* [14]. We simulate 5,000 neurons (using a home-brew implementation) which have a noisy current offset; random input spikes are generated at every time step, with a 20% and 5% probability, for excitatory and inhibitory types, respectively. All synapses are characterised as a 1 ms pulse response; weights for inhibitory synapses are fixed, while excitatory are plastic.

We then look for post-synaptic neuron spikes and collect weight change statistics in a ± 20 ms temporal window in 1 ms time steps. We compute the average change for each time step; Figure 7.14(b) depicts the resulting averages for voltage changes.

We performed a similar experiment using the SpiNNaker implementation and computed the average of generated data points which gives rise to an STDP-like curve (Figure 7.14(c)). A major difference is that the curve gets shifted 1 ms 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 later.

7.3.1 Results

To test the viability of using this learning rule to capture the statistics coming from visual patterns, we set experiments with networks based on a SWTA circuit.

Unsupervised: We first explore an unsupervised learning procedure. The network for this experiment is presented in Figure 7.15 and is composed of an input layer

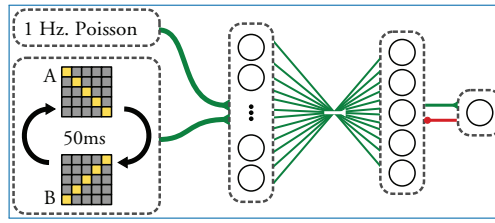

Figure 7.15. SWTA network with visual pattern as input. A 5 × 5 pixel/neuron array is given an input which corresponds to the two main diagonals alternated with a 50 ms delay between them; it is also provided with a 1 Hz noise with a Poisson distribution. This array is connected with plastic connections to 5 target neurons, which in turn are in a SWTA circuit.

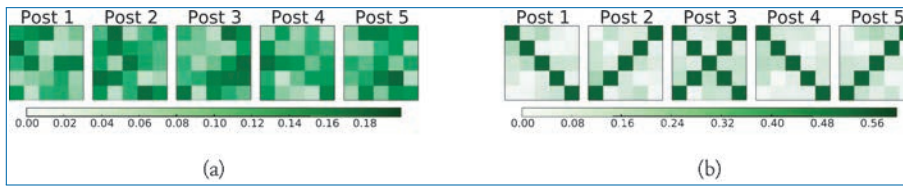

Figure 7.16. Weight changes after alternating pattern simulation. (a) shows the input weights for each target neuron, and these were set at random with a uniform distribution [0.05, 0.2). (b) By the end of the simulation, some neurons have specialised for a pattern as shown by the weights.

and an output layer. The input layer consists of 25 input neurons which serve as a relay for noise and input patterns. The network is given alternating visual patterns (diagonals) as an input; a 1 Hz Poisson noise source is added to the input in order to favour diversity of learning in the output layer. The output layer consists of five neurons which feed a single inhibitory neuron, the latter will reduce the chances of spiking for neurons in the output layer.

The synaptic weights from the input to the output layer were plastic and randomly initialised as shown in Figure 7.16(a). After multiple exposures to the input patterns, the synapses corresponding to the inputs get potentiated (Figure 7.16(b)). Particularly, neurons 1 and 4 become specialised on one pattern while 2 and 5 to the other.

Supervised: To establish a supervised learning regime, we take inspiration from biology and add a special signal which will enhance Long-Term Potentiation (LTP) when present. Biological neurons get further depolarised when N-Methyl-D-aspartic Acid (NMDA) is received and the membrane potential is above a certain level [160]. Implementing a similar behaviour (slow decay and voltage dependence)

on SpiNNaker required altering how input current is computed by default,

$$I = \sum I_+ - \sum I_- \tag{7.17}$$

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 acceptance of current provided by NMDA activation influx, I_ϕ, and thus, Equation 7.17 will be modified to

$$I = \sum I_+ - \sum I_- + \sum I_\phi \tag{7.18}$$

We control the activation of NMDA receptor channels in two ways. The first is through a special ϕ spike which emulates a gating mechanism of the channel (i.e. the presence of both glutamate and glycine [66, 179]). We also use this event to model the current (I_ϕ) created by additional positive ions passing through the opened channel. Secondly, I_ϕ is allowed to pass into the neuron only when the membrane voltage is above the threshold V_ϕ:

$$I_\phi = \begin{cases} I_\phi & \text{if } V_m > V_\phi \text{ or } t - t_\phi < T_\phi \\ 0 & \text{otherwise} \end{cases} \tag{7.19}$$

Furthermore, to mimic the time it takes to close the channels, we added an inertia-like mechanism ($t - t_\phi < T_\phi$ in Equation 7.19) which keeps I_ϕ current flowing for at least T_ϕ simulation steps. To achieve this, we subtract the time at which a ϕ spike arrived (t_ϕ) from the current simulation step and, if the temporal difference is smaller than the inertia window T_ϕ, the current is allowed to keep flowing. To test the supervision mechanism we used a similar network setup to that in the unsupervised case though we need two 'instances' of the network (Figure 7.17); one will 'supervise' the other. Each instance has five output neurons which are each assigned to an input pattern. During the experiment, neurons 1 and 2 of the bottom output population (Figure 7.17) were assigned to learn pattern 1 (forward diagonal), and neurons 3 and 4 were set to learn pattern 2 (back diagonal). The way we induce neurons to learn a particular pattern is to send a ϕ spike 5 ms before the pattern is shown to the corresponding neuron. Neurons in the bottom population connect to the neurons in the top population in a one-to-one manner through the lateral ϕ channel. For this experiment, we used Izhikevich neurons for the output and inhibitory populations which were configured according to the parameters shown in Table 7.4.

Figure 7.18 shows synaptic efficacies at the beginning and end of the training. Each square depicts the values of incoming synapses to a post-synaptic neuron, the top row of squares corresponds to the student population and the bottom row

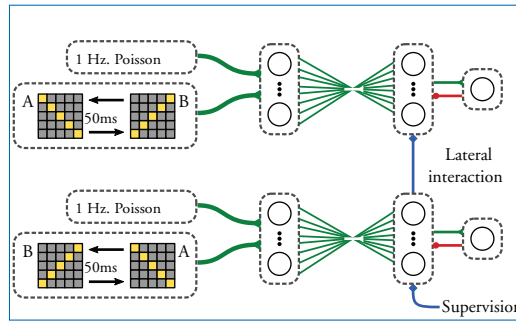

Figure 7.17. Supervision architecture.

Table 7.4. Neuron parameters for pattern learning using ϕ.

	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

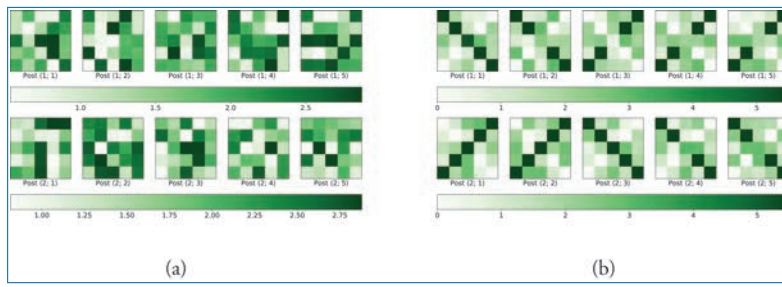

Figure 7.18. Weights at start and end of training using an NMDA-like signal ϕ. (a) Synaptic efficacies are set to random values initially. (b) After ∼30 min of simulation, the weights favour the assigned input patterns.

corresponds to the teacher population. At the start (Figure 7.18(a)), weights are assigned randomly.

During training, a supervision ϕ spike is given to neurons $\langle 2, 1\rangle$, $\langle 2, 2\rangle$, $\langle 2, 4\rangle$ and $\langle 2, 5\rangle$ right before one of the patterns reaches them. Neurons $\langle 2, 1\rangle$ and $\langle 2, 2\rangle$ are assigned the forward diagonal pattern, while neurons $\langle 2, 4\rangle$ and $\langle 2, 5\rangle$ are assigned the backward diagonal pattern. Neuron (2, 3) is allowed to learn its input without any supervision. Since patterns for the student population are delayed (and inverted), the ϕ spikes coming from the teacher enforce neurons in the student population to learn a particular input (backward diagonal for $\langle 1; 1\rangle$ and $\langle 1, 2\rangle$, forward

diagonal for ⟨1, 4⟩ and ⟨1, 5⟩). Figure 7.18(b) shows synaptic efficacies at the end of training, note that the largest weights correspond to the assigned patterns. While the task to learn here is a simple one, the behaviour of the network could be seen as self-supervision and could be applied to networks whose neurons learn parts of a larger problem.

7.4 Neuromodulated STDP

Traditionally, simple models of SNNs have used two types of synapse: excitatory (positive) and inhibitory (negative). These drive changes to the membrane voltage and, indirectly, can produce weight changes [74].

In biological neural networks, there are, in addition, neurotransmitters and neuromodulators that may alter learning processes. Research on dopamine 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.

Furthermore, there are additional cells involved in synapse function – *astrocytes* – which are usually characterised as 'maintainers' in the central nervous system as they keep ionic concentrations stable, form scar tissue on damaged regions and aid energy transfer [231]. Scientists are putting more effort to understand the role of astrocytes as learning modulators [78]. Research shows that these cells are also involved in the regulation of current, frequency, short- and long-term plasticity, and synapse formation and removal [91, 189, 253, 254].

Having a third component modifies Hebbian-based weight updates, and the rule will now depend on the state of three factors (Figure 7.19):

$$\Delta w_{pre,post}(t) \propto h(s(pre), s(post), s(third)) \tag{7.20}$$

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

$$\Delta w_{pre,post}(t) \propto h\left(t, t_{pre}, t_{post}, t_{third}\right) \tag{7.21}$$

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

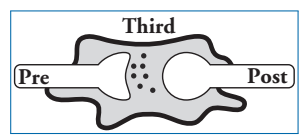

Figure 7.19. Cartoon of third factor interaction on plasticity.

Ponulak and Kasinski [200] introduced an STDP-like rule with a third factor (ReSuMe), in which the extra input is used to get the post-synaptic neuron to spike at a particular time.

When $t_{third} - t_{pre} > 0$, a weight increment (Δ_+) is applied to synapses, whereas a weight depression (Δ_-) is applied when $t_{post} - t_{pre} > 0$. If the post-synaptic neuron activates at the desired time $\left(t_{third} = t_{post}\right)$, depression will be equal to potentiation and the total weight change will be zero.

Gardner and Grüning [70] 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. 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 (≈ 10 ms).

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 that enables learning after the previous activity in the network led to a reward-worthy action [248].

Other modulators (e.g. serotonin and 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 [211].

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

$$\Delta w_{pre,post}(t) \propto g(t, t_{third}) \times f(t, t_{pre}, t_{post}) \tag{7.22}$$

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

7.4.1 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 [59, 75].

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

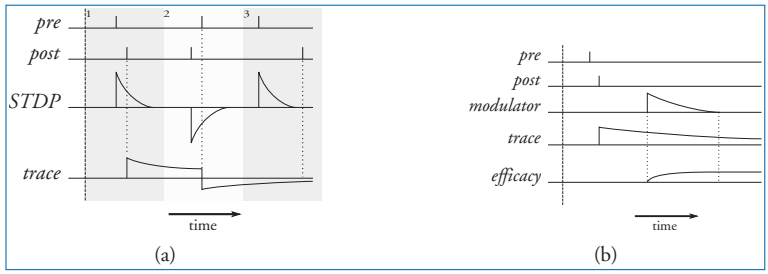

Figure 7.20. Eligibility trace and modulated synaptic efficacy changes.

of the trace [60, 118]. The signal could be dopamine, or another neuromodulator with slower dynamics, decaying on the scale of hundreds of milliseconds.

In Figure 7.20(a), the trace labelled STDP shows the weight change function given the inputs shown in rows *pre* and *post*. Eligibility traces are formed by $\langle pre,\ post \rangle$ spike pairs which are illustrated in zones 1 and 2 in Figure 7.20(a); these cause accumulation of weight changes driven by STDP curves. Since STDP interactions depend on the time at which the pre- and post-synaptic neurons spiked, if these times are sufficiently far apart in time, no weight change is added to the eligibility trace (compare zone 3 with zone 2 in Figure 7.20(a)).

Eligibility traces have much slower dynamics than STDP interactions as illustrated in Figure 7.20(a); the curve in row *trace* decays much slower than any of the curves in row *STDP*. The low decay rate is useful to keep track of how temporally distant weight changes contributed to a particular behaviour.

The modulating neurotransmitter (*modulator* curve in Figure 7.20(b)) 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

$$\Delta weight(t) \propto modulator(t) \times trace(t), \text{ or,} \tag{7.23}$$

$$\frac{dw(t)}{dt} = m(t) \times c(t). \tag{7.24}$$

We implemented the dopamine-based modulated plasticity model as proposed by Izhikevich on the SpiNNaker machine [165]. Eligibility trace dynamics are described by the following equation:

$$\frac{dc(t)}{dt} = -\frac{c(t)}{\tau_c} + STDP(\tau_{-/+})\delta(t - t_{pre/post}); \tag{7.25}$$

where $c(t)$ is the state of the eligibility trace; $STDP(\tau_{-/+})$ is the value from STDP (Figure 7.20(a)) curves and $\delta(t - t_{pre/post})$ is the Dirac delta function. Similarly,

the modulator is governed by

$$\frac{dm(t)}{dt} = -\frac{m(t)}{\tau_m} + M(t)\delta(t - t_{mod}) \tag{7.26}$$

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, and thus, the implemented weight update rule is:

$$\Delta w(t) = \frac{1}{-\frac{1}{\tau_c} - \frac{1}{\tau_m}} c(t_{lc}) m(t_{lm}) \left[e^{-\left(\frac{t-t_{lc}}{\tau_c}\right) - \left(\frac{t-t_{lm}}{\tau_m}\right)} \right]_{t_{lw}}^{t}, \tag{7.27}$$

where lc, lm and lw subscripts indicate the time (event) at which the last eligibility trace, modulator and weight updates were 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 evaluation of the squared brackets in Equation 7.27 is done as in definite integrals.

7.4.2 Credit Assignment

We tested the learning rule replicating an experiment which was originally designed by Izhikevich [117]. The goal is to identify a group of neurons spiking amongst noise and reinforce the groups' connections while keeping other neurons' connectivities in a lower weight range.

The neural network for this experiment consists of 1,000 neurons, divided into two populations. They emit either excitatory or inhibitory signals (*Exc* and *Inh*, respectively). Each neuron connects to others at random with a 10% probability, regardless of the neuron population 'type' (red and green lines in Figure 7.21).

Within the excitatory population, we create groups of 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 10 Hz, and this puts neurons in a biologically plausible setting. In terms of the credit assignment problem, we try to identify a signal (S_1 group activity) in a noisy environment, generated by other groups and random activity incited by background noise.

To search for group S_1's 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 teaching signal.

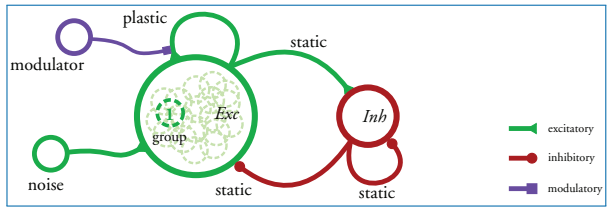

Figure 7.21. Credit assignment experiment network – yet another random balanced network.

Table 7.5. LIF neuron model parameters for the 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

Table 7.6. STDP parameters for the credit assignment experiment.

	A_+	A_-	τ_+	τ_-	τ_c	τ_d
Value	1	1	10	12	1,000	200
Units	–	–	ms	ms	ms	ms

For this experiment, we use standard LIF neurons whose parameters promote higher activity from the elements in the inhibitory group. 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 7.5). We used exponentially decaying, current-based synapses whose temporal constants (τ_{syn_X}) are set to 1 ms to further approximate the original experiment.

The learning algorithm was parametrised so that the area for LTP is 20% greater than the area for Long-Term Depression (LTD) (A_+, A_-, τ_+, τ_- in Table 7.6). Dopamine interaction was characterised in a biologically plausible manner: the temporal constant for the eligibility trace, τ_c, is 1,000 ms, which implies the network will learn events that occurred up to a second ago; the temporal constant for dopamine, τ_d, is 200 ms 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.

The evolution of the average weight in group S_1 is shown in Figure 7.22(a) as a green line, which presents an exponential growth. We can also observe the average

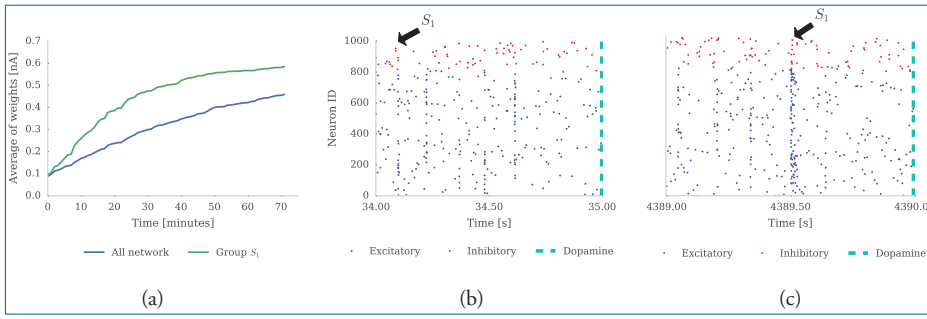

Figure 7.22. Credit assignment experiment network behaviour. Discussed in detail by Mikaitis *et al.* [165].

weight value for every connection in the network as the experiment progresses (blue line), it grows but more slowly and it is expected to stabilize. Spiking behaviour for all groups is similar at the start of the experiment; this can be seen as correlated vertical dots in Figure 7.22(b).

By the end of the experiment, connections which originate from group S_1 are at such a value that most post-synaptic neurons will spike when a neuron in the group is active. This is shown in the middle of Figure 7.22(c) 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.

7.5 Structural Plasticity

Mammalian brains have evolved in an ever-changing environment and, as such, are equipped with a wealth of learning mechanisms. One such mechanism that has been implemented on SpiNNaker is usually referred to as structural plasticity [21]. This mechanism relies on the structural changes in the network guided by some measure. Since SpiNNaker focuses on simulating neurons without morphological detail, structural plasticity is equated with changes in connectivity between neurons – synaptic rewiring.

SpiNNaker is particularly well suited for supporting structural plasticity as a learning mechanism. This software-driven neuromorphic platform is sufficiently flexible to support rewiring in parallel with STDP while still maintaining the real-time wall clock [23]. Furthermore, synaptic rewiring can be used to ensure optimal resource use in a system that is comparatively memory and compute cycle starved [73]. Intuitively, if the rewiring process can be made to maintain a certain synaptic capacity usage while preferentially discarding 'useless' synapses, then the system can

search for the optimal synaptic configuration that fits within the constraints of the system.

SpiNNaker's real-time constraint also means that synaptic rewiring simulations occurring at a slower time scale compared to other neural and synaptic processes can be monitored over longer time scales.

The model chosen for translation onto SpiNNaker was developed by Bamford *et al.* [8]. Their model had some desirable features and posed some interesting challenges. Feature wise, the rewiring rules allowed for synaptic formation and removal driven by Euclidean distance and synaptic weight, respectively.

A certain number of rewiring attempts are performed every simulated second. Each neuron in the network has a limited number of potential synaptic contact points. Attempts either follow the formation or removal rules dependent on the existence of a synapse at a considered contact point.

Formations favour neurons that are relatively close in space, which also represents the first challenge for SpiNNaker. This was the first time that SpiNNaker neural models required spatial information for the simulated neurons. A new, full-strength, connection is formed with a partner neuron that has fired recently if

$$r < p_{form} e^{-\frac{\delta^2}{2\sigma^2_{form}}} \tag{7.28}$$

where r is a random number sampled from a uniform distribution in the interval [0, 1), p_{form} is the peak formation probability, δ is the distance between the two cells and σ^2_{form} is the variance of the receptive field. The result is a Gaussian distribution of formed synapses around the ideal target site, that is, around the target neuron where $\delta = 0$.

If the randomly selected synaptic contact is in use (a synapse already exists), the removal rule is followed. For implementation efficiency and because of the nature of the synaptic plasticity rule (weight-independent STDP), a weight threshold θ_g is selected as half of the maximum allowed weight ($\theta_g = \frac{1}{2} g_{max}$). A synapse is removed if

$$r < p_{elim} \quad \text{where} \quad p_{elim} = \begin{cases} p_{elim-dep} & \text{for } g_{syn} < \theta_g \\ p_{elim-pot} & \text{for } g_{syn} \geq \theta_g \end{cases} \tag{7.29}$$

where r is a random number sampled from a uniform distribution in the interval [0, 1), $p_{elim-dep}$ is the elimination probability used when a synapse is depressed, $p_{elim-pot}$ is the elimination probability used when a synapse is potentiated and g_{syn} is the weight of the synapse under consideration for removal.

The first application of this model of structural plasticity (not to be confused with The Model for Structural Plasticity – MSP – proposed by Butz and van Ooyen

[29]) was to replicate and expand on previous results regarding topographic map formation (Section 7.5.1)

Section 7.5.3 shows that choosing a partner for formation from the set of recently active cells is a powerful mechanism. In the context of MNIST handwritten digit classification, this feature allows for decent accuracy and recall scores in the absence of weight changes as long as the input is encoded using spiking rates. Further, Section 7.5.4 reveals the importance of visualisation in identifying the cause of aberrant behaviour using this particular feature as its main example.

A final application of these synaptic rewiring rules in described in Section 7.5.5 where it is shown to perform elementary motion decomposition after the application of additional minor enhancements.

An alternative formulation based upon information theory, sparse codes and their congruence with recent results about the behaviour of clustered synapses in real dendritic trees is described briefly by Hopkins *et al.* [103].

7.5.1 Topographic Map Formation

A widely observed principle in biological brains is the use of topographic maps, wherein two-dimensional topological (though not necessarily scale) relationships are preserved in projections from one brain region to another.

Neural topographic maps consist of layers of neurons whose reaction to afferent (incoming) stimuli changes with area (Figure 7.23). Such an organisation is characterised by the preservation of neighbour activity from the source to the target layer and provides several advantages in terms of wiring and information processing and integration. Wiring is optimised since neurons generally have limited receptive fields and tend to be interested in spatially clustered locations. As an example, orientation-selective neurons, such as those present in primary visual cortex, are required to have afferents from small regions of the total visual receptive field, thus a topographic organisation ensures that neurons only connect to their immediate neighbours and have limited interaction with those which are further away. More importantly, when neurons form multiple aligned maps, each receiving information from a different modality, they exhibit multisensory facilitation; their response is supra-linear if they receive synchronous stimuli from the same area of space arriving from different modalities. This is the case in the Superior Colliculus, a brain structure that integrates signals from multiple senses and also guides adaptive motor responses [126].

Topographic projections are widespread in the mammalian cortex [123]. Their development has been explored through simulation, with and without spiking neurons, and involving both synaptic plasticity [131, 233, 264] and synaptic rewiring [8]. The latter example has been modelled on SpiNNaker. It is with

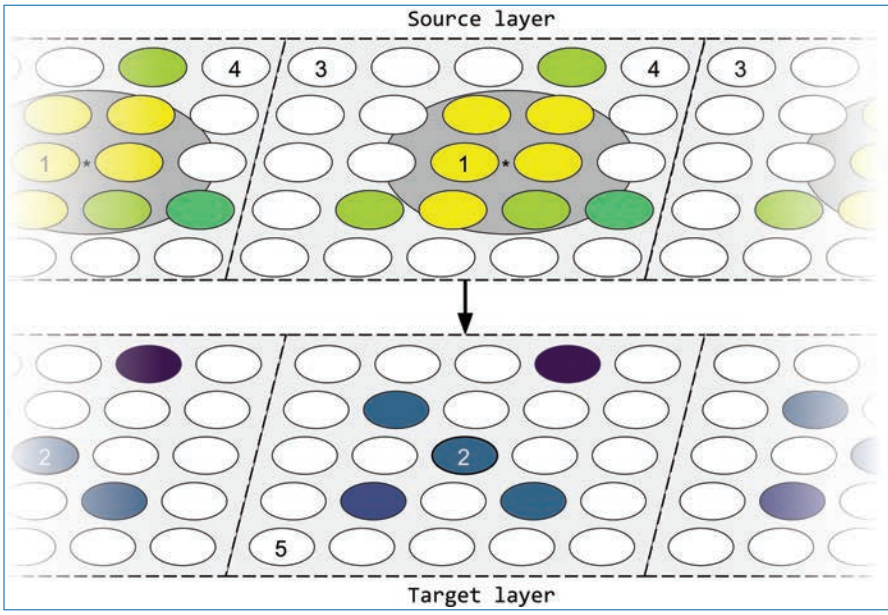

Figure 7.23. Topographic maps. Neuron (2) in the target layer has a receptive field formed by connections from the source layer (feed-forward) as well as connections from within the target layer (lateral). These connections are centred around the spatially closest neuron, that is neuron (1) in the case of feed-forward connections. Connections from more distant neurons are likely to be weaker (indicated by a darker colour).

that model we suggest an architecture capable of handwritten digit classification through supervised learning enabled by the construction of the training architecture.

In short, the suggested model involves the co-operation of two types of mechanisms: activity-independent and activity-dependent mechanisms. The former is represented by the formation rule: it relies on the distance between potential partnering neurons to create a new synapse – spatially clustered neurons will tend to form more connections than neurons that are spatially distant. The latter is composed of two mechanisms: spike-timing-dependent plasticity (STDP) and a removal rule for the synaptic rewiring mechanism. STDP, utilising local spiking information, modifies the weights of synapses connecting neurons together, while the elimination rule preferentially removes those synapses that are depressed – synapses that carry 'useful' patterns or sub-sets of patterns to neurons will tend to be re-inforced, thus are more stable in the long term. Conversely, synapses that usually transmit what amounts to noise will be silenced and are more likely to be pruned. All of the aforementioned mechanisms operate continuously, at a fixed rate, on a population of neurons.

Topographic map quality is assessed using the measures defined by Bamford *et al.* [8]: the spread of the mean receptive field σ_{aff} (grey circular area presented in Figure 7.23) and the Absolute Deviation (AD) of the mean receptive field from its ideal location (the centre of the circular area presented in Figure 7.23 as a star). The former relies on the search for the location around which afferent synapses have the lowest weighted variance (σ^2_{aff}). This is a move away from the centre of mass measurement used by Elliott and Shadbolt [55] for identifying the preferred location of a receptive field. As a result, the centre of the receptive field that is being examined is the location that minimises the weighted standard deviation, computed as follows:

$$\sigma_{aff} = \sqrt{\frac{\sum_i w_i |\vec{p}_{xi}|^2}{\sum_i w_i}} \tag{7.30}$$

where i loops over synapses, x is a candidate preferred location, $|\vec{p}_{xi}|$ is the minimum distance from the candidate location to the afferent for synapse i and w_i is the weight of the synapse. The candidate preferred location x has been implemented with an iterative search over each whole number location in each direction followed by a further iteration, this time in increments of 0.1 units. Thus, the preferred location $\vec{x}$ of a receptive field is given by the function: $\underset{\vec{x}}{\text{argmin}}\, \sigma_{aff}$.

Once the preferred location of each neuron is computed, taking the mean distance from the ideal location of each preferred location results in a mean AD for the projection. We report both mean AD and mean σ_{aff} computed with and without taking into account connections weights. $\sigma_{aff-weight}$ and AD_{weight} are computed using synaptic weights g_{syn}, while $\sigma_{aff-conn}$ and AD_{conn} are designed to consider the effect of rewiring on the connectivity, and thus, synaptic weights are considered unitary.

These metrics are considered in three types of experiments:

- Case 1: STDP and rewiring operating simultaneously in the presence of correlated input
- Case 2: STDP but no rewiring in the presence of correlated input
- Case 3: STDP and rewiring without correlated input

Modelling developmental formation of topographic maps using the presented mechanisms could yield some interesting results with regard to total speed of simulation and could be used to validate whether the network can still generate good-quality topographic maps. Regarding the former, simulations with far more neurons and synapses could benefit from not requiring a fixed connectivity be

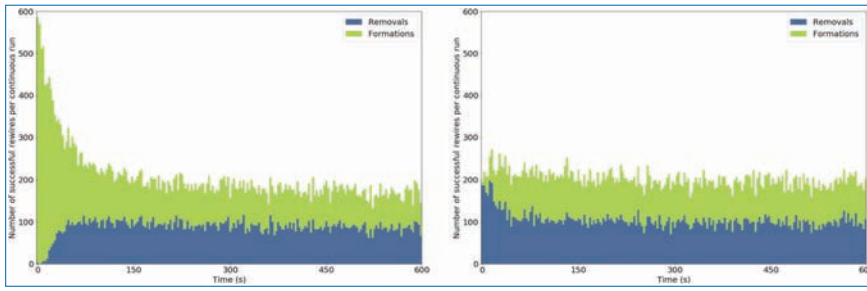

Figure 7.24. Stacked bar chart of formations and removals over time within one simulation. Left: evolution of the network starting from no connections; right: evolution of the network starting from a sensible initial connectivity. The number of formations or removals is aggregated into 3-second chunks.

loaded onto SpiNNaker – the process of interacting with SpiNNaker for loading or unloading data is currently the main bottleneck. The latter is meant as a validation for the model, but also to gain additional insights into its operation, specifically whether maps formed through a process of simulated development react differently to maps that are considered in their 'adult', fully-formed state in need of refinement.

Results from this simulation are presented in tabular form (Table 7.8), with the addition of longitudinal snapshots into the behaviour of the network and mean receptive field spread and drift, as well as a comparison between different initial connectivity types (topographic, random percentage-based and minimal). In the initial stages of the simulation, the σ_{aff} and AD are almost zero due to the lack of connections, but they steadily increase with the massive addition of new synapses. Figure 7.24 shows a side-by-side comparison of the number of rewires between early development and adult refinement. The developmental model initially sees a large number of synapses being formed until an equilibrium is reached at around 10% connectivity. A 10% connectivity is also achieved when starting the network from an adult configuration. This does not mean that every set of parameters will yield the same result. In this case, and all the others in this work, we locked the maximum fan-in for target layer neurons to 32, or 12% connectivity. As a result, the network is bound to have at most that connectivity and at least half that, or 6%, if formation and removal occur with equal probability.

Table 7.8 shows the final, single-trial, results for a network identical to previous experiments, but with a run time of 600 seconds. This ensures the networks have a chance to converge on a value of σ_{aff} and AD. A comparison between Tables 7.7 and 7.8 shows similar results for case 1, but significantly better results for Case 3. These differences are summarised in Figure 7.25. σ_{aff} and AD were computed at the end of three simulations differing only in the initial connectivity: an initial rough topographic mapping as in the previous experiments, a random

Table 7.7. Simulation results presented in a similar fashion to Bamford *et al.* [8] (Table 2) for three cases, all of which incorporate synaptic plasticity. Case 1 consists of a network in which both synaptic rewiring and input correlations are present. Case 2 does not integrate synaptic rewiring, but still has input correlations, while case 3 relies solely on synaptic rewiring to generate sensible topographic maps.

Case	**1**	**2**	**3**
Target neuron mean spike rate	21.15 Hz	20.11 Hz	9.31 Hz
Final mean feedforward fan-in per target neuron	15.91	N/A	11.87
Weight proportion of maximum	0.83	0.72	0.62
Mean $\sigma_{aff-init}$	2.35	2.35	2.35
Mean $\sigma_{aff-fin-conn-shuf}$	2.33	N/A	2.31
Mean $\sigma_{aff-fin-conn}$	1.62	2.35	1.85
p(WSR $\sigma_{aff-fin-conn}$ vs. $\sigma_{aff-fin-conn-shuf}$)	2.80×10^{-43}	N/A	3.65×10^{-27}
Mean $\sigma_{aff-fin-weight-shuf}$	1.61	2.32	1.78
Mean $\sigma_{aff-fin-weight}$	1.49	1.92	1.57
p(WSR $\sigma_{aff-fin-weight}$ vs. $\sigma_{aff-fin-weight-shuf}$)	4.03×10^{-33}	4.02×10^{-43}	1.44×10^{-21}
Mean AD_{init}	0.81	0.81	0.81
Mean $AD_{fin-conn-shuf}$	0.82	N/A	1.09
Mean $AD_{fin-conn}$	0.77	0.81	0.91
p(WSR $AD_{fin-conn}$ vs. $AD_{fin-conn-shuf}$)	0.39	N/A	0.002
Mean $AD_{fin-weight-shuf}$	0.79	0.92	1.04
Mean $AD_{fin-weight}$	0.85	0.79	1.07
p(WSR $AD_{fin-weight}$ vs. $AD_{fin-weight-shuf}$)	0.0002	0.0001	0.58

10% initial connectivity balanced between feedforward and lateral, and almost no initial connectivity (in practice, one-to-one connectivity was used due to software limitations). We do not simulate the case without synaptic rewiring, as the results would be severely impacted by the lack of rough initial topographic mapping. The

Table 7.8. Results for modelling topographic map formation from development (minimal initial connectivity).

Case	**1**	**3**
Target neuron mean spike rate	18.49 Hz	9.80 Hz
Final mean fan in / target neuron	17.69	11.92
Weight proportion of maximum	0.83	0.63
Mean $\sigma_{aff-init}$	0	0
Mean $\sigma_{aff-fin-conn-shuf}$	2.39	2.30
Mean $\sigma_{aff-fin-conn}$	1.56	1.67
p(WSR $\sigma_{aff-fin-conn}$ vs. $\sigma_{aff-fin-conn-shuf}$)	1.14×10^{-43}	2.27×10^{-35}
Mean $\sigma_{aff-fin-weight-shuf}$	1.56	1.59
Mean $\sigma_{aff-fin-weight}$	1.44	1.26
p(WSR $\sigma_{aff-fin-weight}$ vs. $\sigma_{aff-fin-weight-shuf}$)	1.25×10^{-36}	6.21×10^{-31}
Mean AD_{init}	0	0
Mean $AD_{fin-conn-shuf}$	0.79	0.99
Mean $AD_{fin-conn}$	0.64	0.85
p(WSR $AD_{fin-conn}$ vs. $AD_{fin-conn-shuf}$)	1.3×10^{-4}	0.01
Mean $AD_{fin-weight-shuf}$	0.66	1.02
Mean $AD_{fin-weight}$	0.65	0.96
p(WSR $AD_{fin-weight}$ vs. $AD_{fin-weight-shuf}$)	0.84	0.51

final mean value of AD is improved in both experiments involving initial non-topographic connectivity and σ_{aff} is improved when the network starts with no initial connectivity.

7.5.2 Stable Mappings Arise from Lateral Inhibition

Following the previous results, coupled with the tendencies exhibited by the results generated for Case 3 in Figure 7.26, further simulations have been run to confirm whether input correlations are necessary to generate stable topographic maps. Our results are inconclusive as although the weighted AD of the mean receptive field increased to 2.25, or 1.64 if only considering connectivity, the final

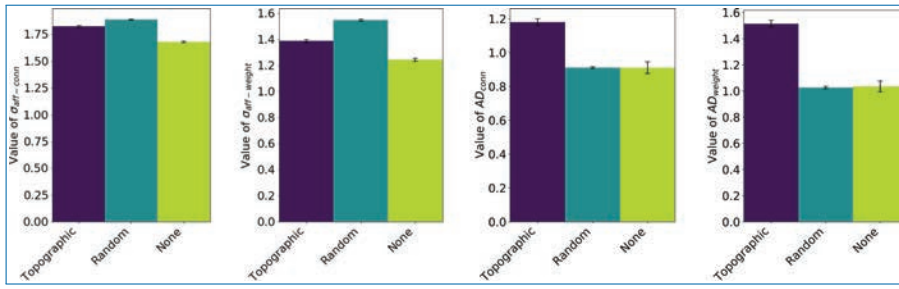

Figure 7.25. Comparison of final values for σ_{aff} and AD in the case where input correlations are absent (Case 3). Three types of networks have each been run 10 times (to generate the standard error of the mean), each starting with a different initial connectivity: an initial rough topographic mapping as in the previous experiments, a random 10% connectivity (5% feedforward, 5% lateral) and almost no connectivity (one-to-one connectivity used due to software limitations).

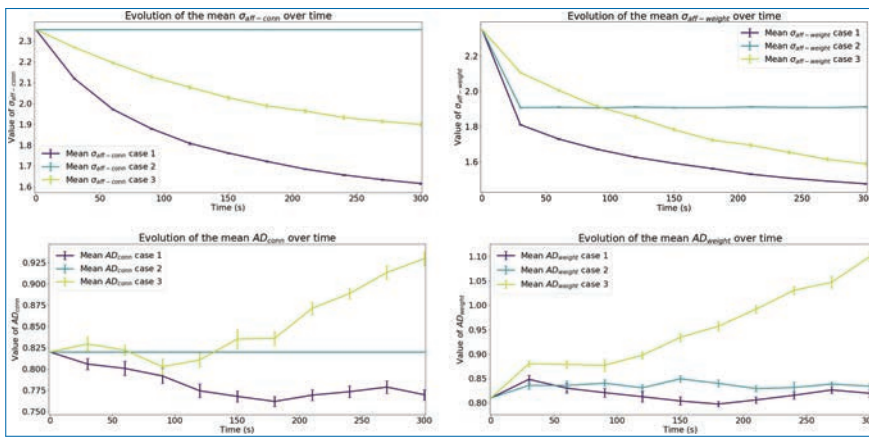

Figure 7.26. Evolution of results of interest. The top row shows the evolution of the mean spread of receptive fields over time, considering both unitary weights ($\sigma_{aff-conn}$) and actual weights ($\sigma_{aff-weight}$) at that point in time. The bottom row shows the evolution of the mean absolute deviation of the receptive fields considering connectivity (AD_{conn}) and weighted connectivity (AD_{weight}). Error bars represent the standard error of the mean.

mean number of feedforward synapses reduces to around 5, making these statistics unusable.

We hypothesise that since the feedforward is reduced so heavily, both in terms of connectivity (on average 5 incoming connections for each target-layer neuron) and in terms of synaptic weights (0.3 of maximum possible for the present connectivity) that the lateral connections within the target layer drive the comparatively high activity in the target layer (Figure 7.27). This, in turn, causes the pre-synaptic

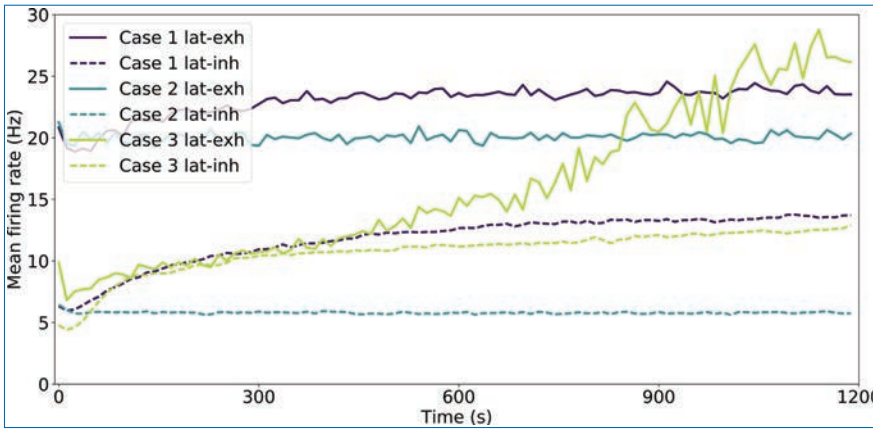

Figure 7.27. Target layer firing rate evolution throughout the simulation. The instantaneous firing rate has been computed in 1.2-second chunks for simulations where lateral connections are excitatory (lat-exh) and for simulations where lateral connections are inhibitory (lat-inh).

partner selection mechanism to focus its attention mostly on the target layer. We have achieved a reduction in the target layer firing rate by introducing inhibitory lateral connections. This is sufficient to generate a stable topographic mapping that matches quite closely the results of the original network when both input correlations and synaptic rewiring are present: $\sigma_{aff-conn} = 1.74$, $\sigma_{aff-weight} = 1.38$, $AD_{conn} = 0.85$, $AD_{weight} = 0.98$; all results are significant. The combined choice of sampling mechanism and lateral inhibition has a homeostatic effect upon the network.

Conversely, in the cases where input correlations are present, we see stable topographic mapping, regardless of the presence of synaptic rewiring, as well as significantly more feedforward synapses. Finally, no applicable network was negatively impacted by initialising the connectivity either randomly or with minimal connections.

To sum up, the model can generate transiently better topographic maps in the absence of correlated input when starting with a negligible number of initial connections or with completely random connectivity. These results can also be stabilised with the inclusion of lateral inhibitory connections preventing self-sustained waves of activity within the target layer. Experiments using correlated inputs do not require inhibitory lateral feedback either for reducing the spread of the receptive field or for maintaining a stable mapping. Finally, the model has proven it is sufficiently generic to accommodate changes in initial connectivity, as well as type of lateral connectivity, that is, the change from excitatory to inhibitory synapses.

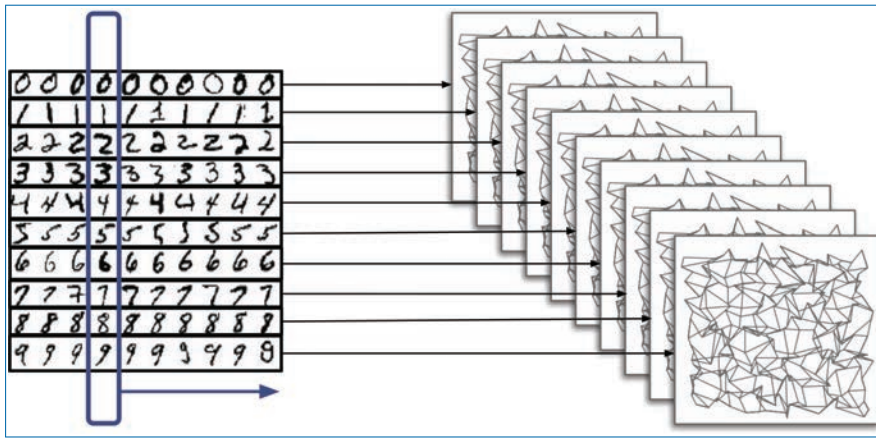

Figure 7.28. Network architecture used for training. A source layer displays a series of examples of handwritten digits; each example from a particular class is projected to the target layer corresponding to that class.

7.5.3 MNIST Classification in the Absence of Weight Changes

With an established model for synaptic rewiring [23], it is now possible to perform classification tasks using a simple architecture [103].

The model is equivalent to the supervised learning paradigm in ANNs. Data are labelled using a dedicated projection from a source layer to the corresponding target layer. A layer of neurons providing examples belonging to a class connects exclusively to a population which learns to recognise members of that class. Figure 7.28 shows the network architecture of the training regime, where each source in a source-target population pair displays a digit for 200 ms for a combined total simulation time of 300 seconds; the initial connectivity between each source-target pair is 1%.

To generate the input, each original digit is filtered via convolution using a 3×3 centre-surround kernel, mimicking the response of the highest resolution retinal ganglion cells. The kernel is normalised to sum to zero with an auto-correlation equal to one. Finally, a threshold is applied after the convolution operation, resulting in edge detection. Transmission within the network is achieved through the use of neurons generating Poisson spike trains; each pixel within the 28×28 image is mapped to two Poisson neurons, one for the on channel and one for the off channel. The top of Figure 7.29 shows examples of input digits before adding background noise; all of the feed-forward connections are excitatory.

The bottom image in Figure 7.29 shows that it is possible to identify visually what each target layer has learnt; time-averaged digits from each class are embedded

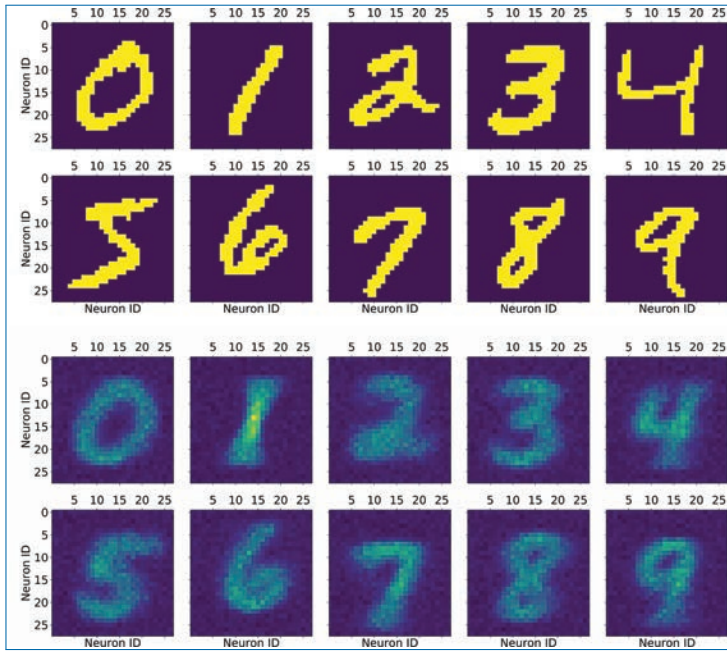

Figure 7.29. Top: Input rate-based MNIST digit representation. Bottom: Reconstruction of the learned digits when connectivity is adapted using only structural plasticity.

into the connectivity of the network. It is then possible to test the quality of classification. For this, we make use of a single source layer. The previously learnt connectivity is used to connect all of the target layers to the source layer, and all plasticity is disabled. The source layer now displays class-randomised examples, each for 200 ms. The classification decision is made off-line, based on which target layer has the highest average firing rate within the 200 ms period.

This is not a state-of-the-art MNIST classification network (it achieves a modest accuracy of 78% and a Root Mean Squared Error (RMSE) of 2.01, Figure 7.30 reveals what mistakes the network made) as each input digit class is represented only as an average for that class, but it serves here to demonstrate that synaptic rewiring can enable a network to learn, unsupervised, the statistics of its inputs. Moreover, with the current network and input configuration, the quality of the classification is critically dependent on the sampling mechanism employed in the formation of new synapses. Random rewiring, as opposed to preferentially forming connections to neurons that have spiked recently, could achieve accurate classification only if operating in conjunction with STDP.

Finally, this approach is also critically dependent on the encoding scheme used to represent the input. A rate-based encoding as shown here is required if no STDP is present.

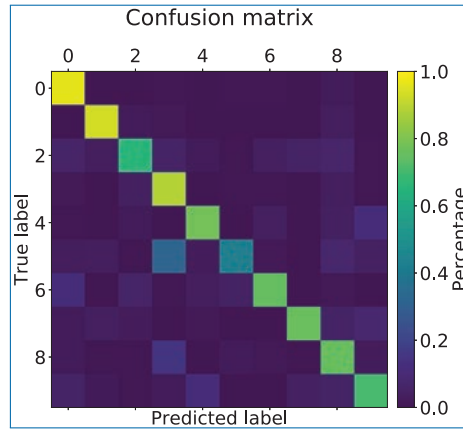

Figure 7.30. Classification confusion matrix.

7.5.4 Visualisation, Visualisation, Visualisation

While working on implementing such an alien mechanism on SpiNNaker, it was important to ensure that everything behaved correctly. It took more than a year to reach this goal. More than a year of simulations, trial and error, blood and sweat, printing parameters from SpiNNaker and saving them to disk, analysing and re-analysing. Frustratingly, we were stuck at debugging weird behaviour revealing itself as unexpected stripes in the connectivity matrix. Jupyter notebooks played a central role in this process, although in hindsight they should have been phased out after the initial first complete run. This particular issue had evaded all attempts at a systematic sweep of parameters and histogram plotting. Until one day…

Discipline is of course important in identifying bugs – this is why we can point at the exact simulation results that flagged the issue.[1] However, I believe that just as important as discipline can be a singular plot. In our case that plot is shown in the middle of Figure 7.31. It perfectly captures our issue: a systematic bias in the choice of pre-synaptic partner caused by the sub-millisecond ordering of spikes as they are generated by the input Poisson spike source.

The problem: we were selecting the last neuron to have fired as a partner for formation, thus ordering the spikes generated within a millisecond time step. The solution: correctly appreciate that because of the resolution of the time step, the input spikes were simultaneous and the source of one should be selected at random as a partner for rewiring.

To conclude, the solution only emerged after seeing a plot that would later be included in a paper by George *et al.* [73]. It looked at a vector of neurons that

1. http://dx.doi.org/10.17632/xfp84r5hb7.1#folder-36833daa-91a8-499c-a898-65a96e22958b

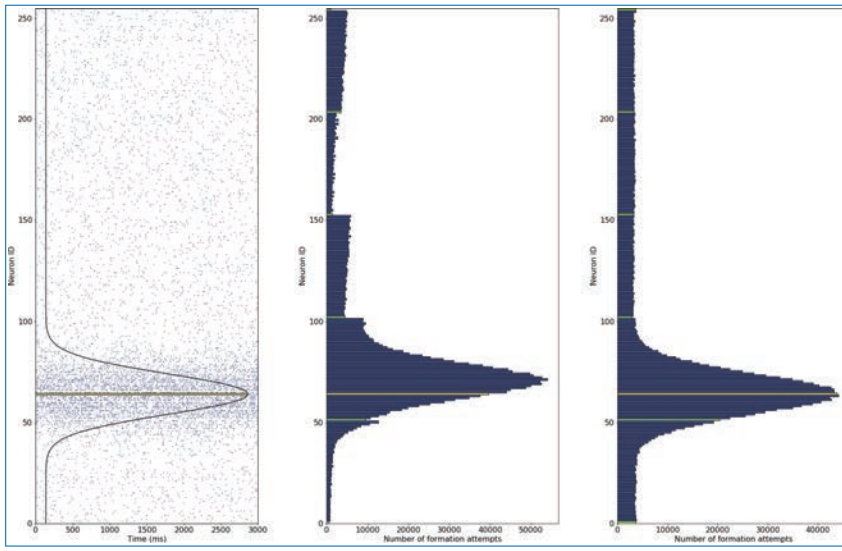

Figure 7.31. Left: the input spiking activity represented by a Poisson spike generator with rates described by the Gaussian curve in black; centre: the neuron identifiers considered for formation throughout an entire simulation if this choice relies on selecting the last neuron to have spiked; right: the neuron identifiers considered for formation throughout an entire simulation if the choice relies on selecting an arbitrary partner among the ones that have spiked since the last time step. In bright yellow, aligned across all plots: the neuron with the highest firing rate. Additionally, the partitioning of the pre-synaptic population is highlighted in green in the central and right-most figures.

generated input spikes with bimodal firing rates and showed how the connectivity had adjusted. We needed only look at which sources the formation attempts were considering to reveal the issue.

7.5.5 Rewiring for Motion Detection

We build on previous work [103] and present an end-to-end approach to perform elementary motion decomposition using LIF neurons and structural and synaptic plasticity [22]. Further, the computational platform which is the basis for these simulations is event-driven [207], including the spiking visual input provided to the network. The biologically inspired sensory processing method presented here is an alternative to traditional frame-based computer vision.

We show that (1) the presented architecture allows for unsupervised learning; that (2) synaptic rewiring enhanced to initialise synapses by drawing from a distribution of delays produces more specialised neurons; and that (3) a pair of readout neurons is sufficient to correctly classify the input based on the target layer's activity

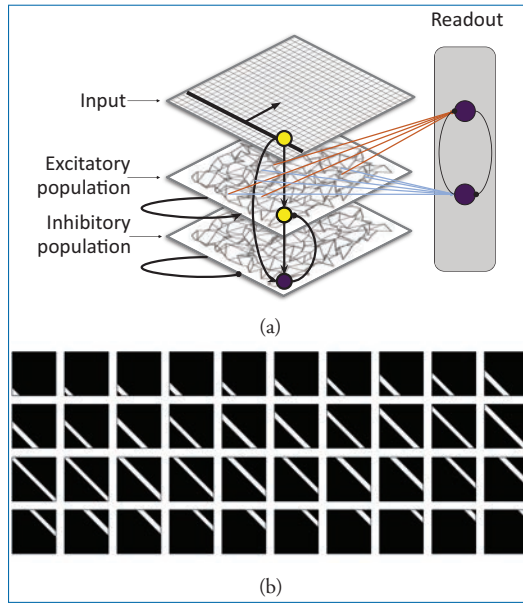

Figure 7.32. (a) Network architecture. (b) Example input 45° movement represented as its constituent frames (before processing to generate spikes). A new frame is presented every 5 ms and, in total, the presentation of an entire pattern takes 200 ms.

using rank-order encoding (first classification neuron to spike wins), rather than spike-rate encoding (classification neuron that fires most in a time period wins).

The SNN architecture (pictured in Figure 7.32(a)) is designed to allow unsupervised learning through self-organisation using synaptic and structural plasticity mechanisms [23]. Neurons in the two target populations are modelled as being positioned at integer locations on a 32 × 32 grid with periodic boundary conditions. The excitatory population contains neurons that receive sparse excitatory connections from the input layers and from themselves, while projecting to the inhibitory layer and to the readout neurons responsible for the final motion classification decision. The inhibitory population follows a similar structure, but only projects using inhibitory synapses. Very strong inhibition is also present between the readout neurons, implementing a WTA circuit. The networks are described using the PyNN simulator-independent language for building neuronal network models [44] and the SpiNNaker-specific software package for running PyNN simulations (sPyNNaker [207]).[2] The model is simulated in real time on the SpiNNaker many core neuromorphic platform using previously presented neuron and synapse dynamics [23].

2. The data and code used to generate the results presented here are available from doi: 10.17632/wpzxh93vhx.1

The input stimulus consists of bars encoded using spikes representing 'ON' and 'OFF' pixels (see Figure 7.32(b) for an example before filtering using a previously described technique [194]) as well as a background level of Poisson noise (5 Hz). Each stimulus is presented over a 200 ms time period always moving at a constant speed (200 frames per second). During training, the target layers are presented with bars moving in two directions (Eastward or at 0° and Northward or at 90°), but during testing they are presented with moving bars in all directions (randomised over time, in 5° increments) – weights and connectivity are fixed during this latter phase. The simulations are initialised with no connections and are trained for around 5 hours, while testing occurs over 20 minutes. As a result of the chosen testing regime, the networks sees over 80 moving bar presentations at each of the 72 angles. This allows us to perform a *pair-wise independent t-test* between the responses at each of the angles in the two cases and establish whether their responses are statistically different. The readout neurons are trained and tested separately from the rest of the network – this process takes on the order of a minute.

Using the structural plasticity mechanism implemented for SpiNNaker, new synapses are formed in two regimes: with heterogeneous, random delays ([1, 15] ms, uniformly drawn) and homogeneous, constant (1 ms) delays; the latter is taken to be the control experiment. Further, according to the structural plasticity mechanism, depressed synapses are more likely to be removed. This optimises the use of the limited synaptic capacity available for each post-synaptic neuron [73]; neurons have a fixed maximum fan-in of 128 synapses with fixed delays.

The Direction Selectivity Index (DSI) will be computed for each neuron after training: $DSI = (R_{pref} - R_{null})/R_{pref}$, where R_{pref} is the response of a neuron in the preferred direction and R_{null} is the response in the opposite direction [157]. We compute it for each of the possible directions and establish the preferred direction as that which maximises the DSI. Individual neurons generally have noisy responses. As such, to avoid the noise skewing the computation, we filter the response of the neuron by applying a weighted average on individual angle responses.

More formal analysis, although less specific to the task of motion detection, is also performed. Entropy is computed using each neuron's normalised spiking profile. After testing, each neuron's firing profile $R(X)$ is computed in relation to each one of the $i = 1 \rightarrow 72$ input movement directions. Normalisation is performed by dividing every response by the sum of all responses:

$$P_i(X) = R_i(X) \sum_j R_j(X) \tag{7.31}$$

so that $\sum_i P_i = 1$ for each neuron X. The firing profile of the neuron can thus be interpreted as the neuron's confidence in the input movement direction. We can

compute the entropy of a neuron's response:

$$H(X) = -\sum_i P_i(X) \log_2 P_i(X) \tag{7.32}$$

The maximum entropy in the presented system is thus $-\log_2(1/72) \approx 6.17$ bits, which is equivalent to neurons displaying equal spiking activity in all presented angles, or no selectivity whatsoever. Neuron X is said to be very selective if simultaneously maximises DSI ($DSI(X) \rightarrow 1$) and minimises entropy ($H(X) \rightarrow 0$). It is sufficient for a neuron to have $DSI(X) \geq 0.5$ to be considered selective.

Both DSI and entropy are used to select and investigate the behaviour of individual neurons. In the following section, we will display quadruplets of individual neuron responses that have maximal grid-aligned responses and minimal orthogonal and opposite responses: $argmax_X = R_i - (R_{i+90^\circ} + R_{i-90^\circ} + R_{i+180^\circ})$. Here i is in turn: 0°, 90°, 180° and 270°, that is, the cardinal directions. The distributions of entropy and DSI are also included for all experiments. Comparison of these distributions is performed by applying both Welch's t-test and Kruskal's h-test.

After training the readout units, it is possible to establish class inputs. Based on the predicted label and the known true labels, we report the accuracy, recall F-score of the network, as well as the RMSE. We define Tp and Tn to mean the number of true positive and true negative examples, while Fp and Fn refer to the number of false positives and negatives, respectively. Recall or sensitivity, intuitively the ability of the classifier to find positive samples, is reported as $Tp/(Tp+Fn)$. Precision or the positive predictive value, intuitively the qualitative ability of the classifier not to label as positive an example that is negative, is computed as $Tp/(Tp + Fp)$. Given these metrics, we can now compute the weighted average between precision and recall to generate the F-score $F1 = 2\,(\text{precision} \times \text{recall})/(\text{precision} + \text{recall})$. Due to the readout architecture and experimental parameters, we also investigate the number of instances in which no readout neuron produces a spike.

The response of the excitatory population in each regime (incorporating heterogeneous delays or not) is plotted for each testing direction (minimum, mean and maximum responses presented in Figure 7.33(a)). The polar plot reveals the firing rate (Hz) of neurons during testing when the input is moving in each of the 72 directions from 0° to 355° in 5° increments in a random order. The network response shows that neurons are responding preferentially to movement, rather than simply to the shape of the input, because the response is asymmetrical – it can differentiate between, for example, a vertical bar moving eastward and the same vertical bar moving westward. The *pair-wise independent t-test* is performed to compare the network response in the two regimes (Figure 7.33(c), red line signifies that $p \geq 0.001$ for that particular angle); the response is higher in one training direction

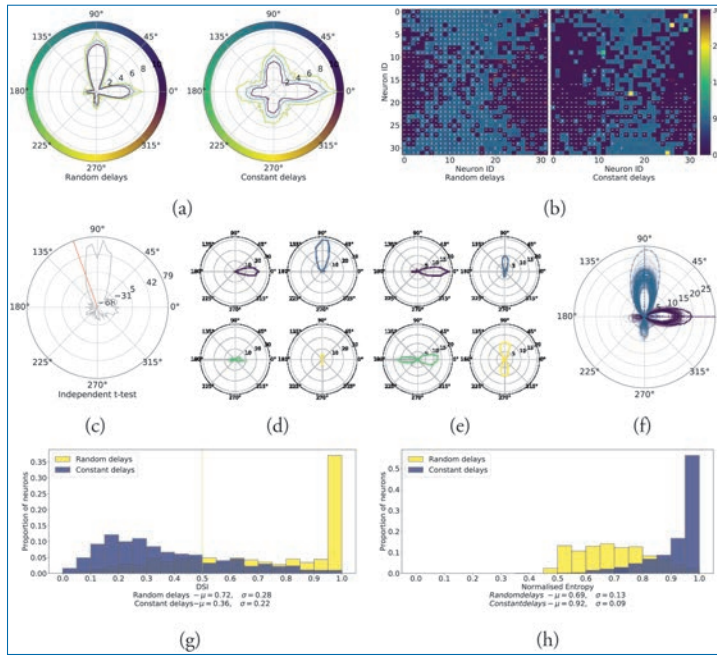

Figure 7.33. Spiking activity comparison between networks where rewiring assigns random and constant delays, respectively, to new synapses. Both networks were trained using bars moving eastward (0°) and northward (90°). (a) the minimum, mean and maximum aggregate excitatory population firing response (Hz); (b) neuron angle preference based on maximum firing rate encoded by the colour and DSI represented by the arrow direction (it is only present if DSI $\geq$0.5); (c) pair-wise independent t-test comparing the network with heterogeneous delays (on the left in a and b) to the control setup (on the right in the same subplots) – red lines show the angle at which the comparison yielded insignificant results; (d) selected individual neuron responses in the 4 grid-aligned directions (random delays); (e) selected individual neuron responses in the 4 grid-aligned directions (constant delays); (f) individual overlaid selective neuron responses after filtering (DSI $\geq$0.5); (g) histogram comparison of all DSI values in the two networks; (h) histogram comparison of all entropy values in the two networks.

(90°) and less in the other (0°) for the network with heterogeneous delays compared to the control. As such, we proceed by examining individual neurons rather than the average network behaviour. The spatial organisation of neurons and their preferred angle is presented in Figure 7.33(b), showing that local neural neighbourhoods become sensitised to the same input statistics. There we also look at neurons' maximum responses (encoded by the colour of the cell) in conjunction with the direction that maximises DSI (arrow direction) and DSI $\geq$ 0.5 (arrow presence). The DSI histogram presented in Figure 7.33(g) compares the two networks; the

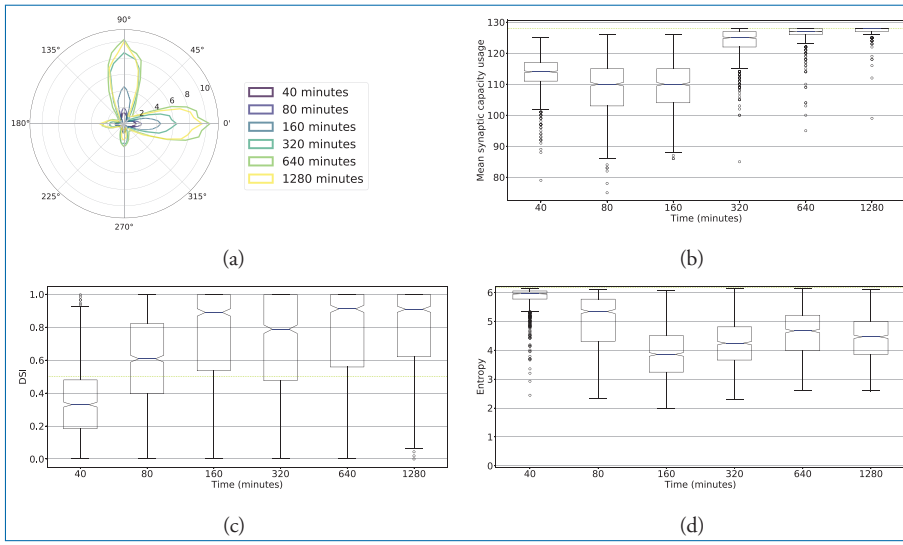

Figure 7.34. Network evolution over a wide range of simulation run times when trained on two angles. (a) average network firing response during inference when trained for ever increasing times; (b) average number of afferents (incoming connections) for each neuron in the excitatory target layer; (c) DSI distribution displayed as a *boxplot* for each simulation in (a); (d) entropy distribution displayed as a *boxplot* for each simulation in (a). Note: Each data point is a different simulation.

control network has significantly fewer selective neurons (251 compared to 744) and selectivity is lower on average. Individual responses of our simulated neurons resemble the direction selectivity found in Superior Colliculus [112].

Further, we examine the network behaviour over a wide range of simulations times, ranging from 40 minutes up to 20 hours. Figure 7.34(a) shows the evolution of the population-level firing rate and the evolution of the DSI metric (Figure 7.34(c)). The network is thus shown to be stable over long periods of time, rather than showing destructive dynamics.

A readout or classification mechanism relying on two mutually inhibitory neurons is sufficient to resolve the two directions presented in the input. Static excitatory connectivity originating from the excitatory layer results in a potential 100% classification accuracy based on rank-order encoding. After 40 seconds, the two neurons have self-organised to respond to one of two input patterns. Figure 7.35 shows the spiking behaviour of the two neurons in the first 1.8 seconds of training and testing. STDP reduces the latency in neural response to the stimuli, making the neurons respond to the stimulus onset, thus making them ideal for classification using rank-order encoding, rather than a WTA classification based on spike count across a time period [103].

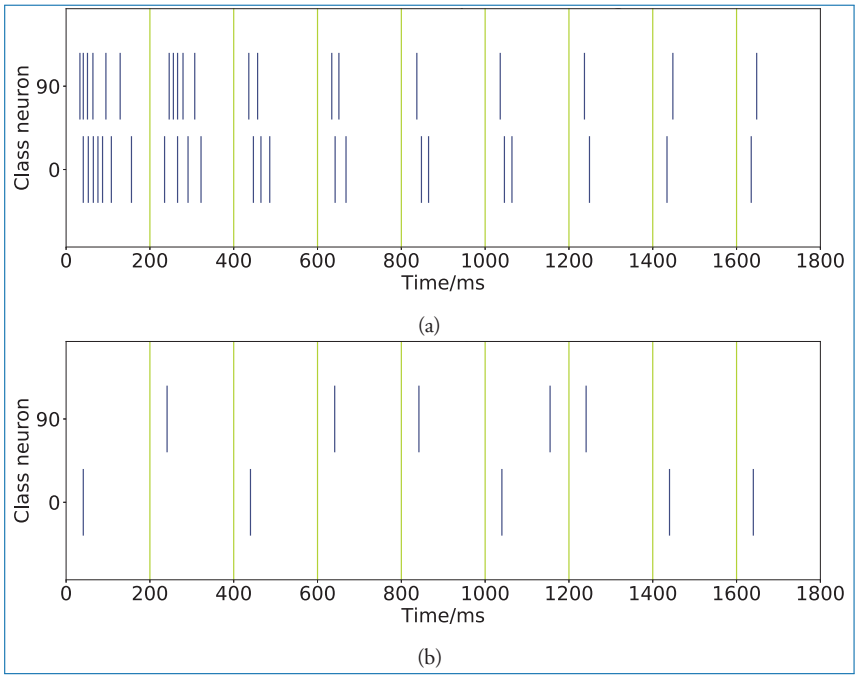

Figure 7.35. Initial spiking activity of the two readout neurons during training (a) and testing (b). The full-height vertical bars denote the edges of the pattern presentation time bins (every $t_{stim} = 200$ ms). Neuron class is established *post hoc* as the one maximising classification accuracy.

7.6 Neuroevolution

SNNs are not solely defined by hierarchical-layered architectures and activation functions but also by neuron model parameters that can alter neuron behaviour over time. The extra spatio-temporal degrees of freedom afforded by SNNs give models a larger parameter space than ANNs. This is of particular note for SNNs that are models of biological neural networks where experimental or ethical limitations mean that it can be difficult or impossible to collect the data *in vivo* to define all the necessary model parameters. Optimisation techniques can be used to explore the parameter spaces of such models and find plausible values. We anticipate that this will be a fruitful area of future research.

A practical method for optimisation and the exploration of the large search spaces seen with SNNs is to use gradient-free (also known as random search) optimisation methods such as Evolutionary Algorithms (EAs). EAs involve evaluating a population of potential solutions (known as agents or individuals) against a

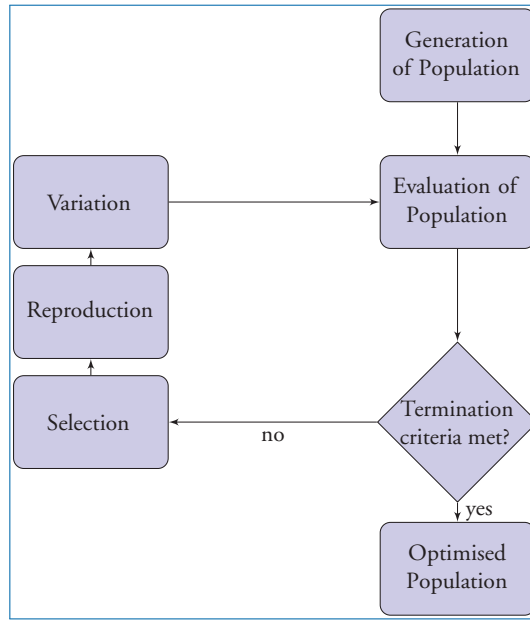

Figure 7.36. A flowchart showing the steps in an EA.

target task.[3] The agents in the population that perform best are selected to reproduce, with the offspring having some variation applied. The offspring form the population for the next generation, see Figure 7.36, often with a small portion of the best-performing agents automatically passing into the next generation. This is known as elitism and has been shown to aid convergence to solutions [46]. Like evolution by natural selection in biology [43], survival of the fittest in EAs is an unguided force which can lead towards better performing agents.

How the performance of agents is evaluated is task specific; however, it does not require gradient information, making an EA approach particularly well suited for tasks in which the error is either difficult or impossible to differentiate. The terms EA and Genetic Algorithm (GA) are often used interchangeably but technically GAs are a subclass of EAs in which agents are encoded as discrete values in 'genes' and random mutation and crossover adding variation to offspring.

The scale of large SpiNNaker systems, such as the one million core machine at the University of Manchester, lend themselves to population-based search methods. The parallelism available with such systems allows model execution to become invariant with respect to population or network size, the main components leading

3. So far, population has been used in the context of PyNN to describe a group of neurons. In the context of EAs population is used to refer to a group of individuals (whole networks) to be optimised, unless otherwise specified.

to increased simulation time on other platforms. Larger population sizes increase the effectiveness of an EA as they increase the coverage of the search space. With more serial systems, a practical limit is placed on the size of the population but with SpiNNaker such scaling problems are diminished. There is an overhead in terms of loading models on to SpiNNaker; however, by separating the population of networks into a collection of smaller jobs, the loading time can be made similar to the time taken to run a single network assuming sufficient auxiliary computational resource.

7.6.1 Pac-Man on SpiNNaker

The first concrete application of neuroevolution to develop SNNs on SpiNNaker was undertaken by Vandesompele *et al.* [259] in which the NeuroEvolution of Augmenting Topologies (NEAT) algorithm [237] was used to generate an agent able to play the Pac-Man arcade game as well as solve the XOR problem. In Pac-Man, the agent had a view of the tiles one or two spaces around it, closer to modelling an agent in a maze than a conventional human player. The algorithm converges on the XOR problem, although it required 49,750 evaluations compared to the 13,459 in the worst-case performance of NEAT using an ANN (average of 4,800). The Pac-Man game was a proof of concept. The performance did improve with time; however, it did not get close to the theoretical maximum score. This is to be expected of an agent that cannot see further than two positions around itself as any high level of forward planning necessary to eat ghosts, representing a large increase in score, is not feasible.

7.6.2 Further Exploration of NEAT

Work is currently being undertaken to increase the efficiency of the algorithmic implementation of NEAT on SpiNNaker. One key aspect to improving performance is to run the target task on SpiNNaker directly so that the model and the environment with which it interacts need not communicate via Ethernet; instead of having spike communication with a host computer, connections are kept local and fast, eliminating an input/output bottleneck and increasing the scale of population possible with SpiNNaker. So far, models running on SpiNNaker have been created for the multi-armed bandit task, inverted pendulum and Breakout (the classic Atari game) as well as a number of other tasks.

7.6.3 An Evolutionary Optimisation Framework for SpiNNaker

Another approach developed an EA framework for SNN models running on SpiNNaker and looked to understand the scaling and performance of this kind

of optimisation. The weight parameters of a small convolutional network for the MNIST digit recognition task were evolved as a test case. A GA was chosen as the optimisation algorithm because of its biological relevance and the potential for model evaluation to be parallelised by running multiple models simultaneously on SpiNNaker.

7.6.4 Methods

Figure 7.37 shows the structure of the simple convolutional SNN model, the weights of which were optimised for the MNIST digit recognition task using a GA. The spikes of the 28 × 28 input layer were rate-coded representations generated from the MNIST images. The hidden layer was a 24 × 24 layer that is the convolution of input with a 5 × 5 filter. The hidden layer was fully connected to the 10 output neurons. The 25 weights of the filter and the 5,760 weights of the fully connected layer were encoded in a 5,785 base gene, with the bases taking integer values in the range −1 to +1. The details of the GA used are detailed in Table 7.9. Two experiments were carried out to better understand the effect of different initialisation on the evolution of the population over 304 generations: the

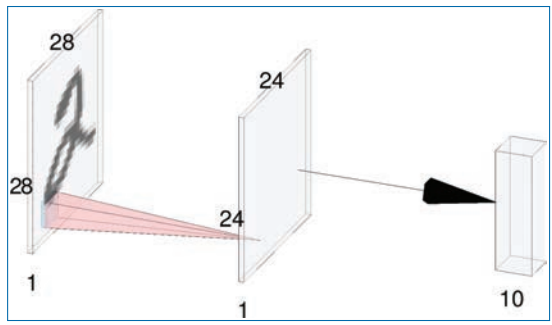

Figure 7.37. The structure of the simple SNN model optimised using a GA.

Table 7.9. A summary of the GA parameters.

Variable	Value
Population size (individuals)	24,000
Mutation rate	0.1%
Mutation type	Base substitution
Crossover rate	50%
Crossover type	Two-point crossover

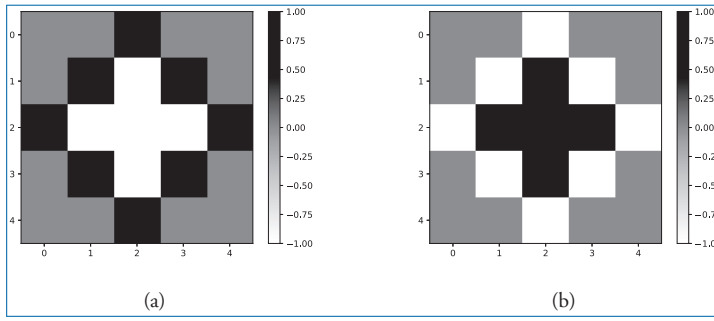

Figure 7.38. The centre-surround filters used to seed the seeded population.

first experiment tracked the accuracy of a population of 24,000 individuals with randomly initialised genes (the unseeded population) and in the second the initial population (the seeded population) was seeded with 12,000 individuals with one centre-surround filter (6,000 positive, 6,000 negative, see Figure 7.38). In each of these experiments, 7.29×10^6 networks were evaluated on SpiNNaker.

7.6.5 Results

Figure 7.39 shows the evolution of the training accuracy of the two populations over 304 generations. The five top performing individuals from the final populations were evaluated against the MNIST testing set and the best individuals gave 66.7% and 63.9% testing accuracy, unseeded and seeded, respectively. These results are far from state-of-the-art accuracies but demonstrate that a GA can be used for optimisation in this way.

It was hypothesised that the same convolutional filter may evolve from an unseeded population independent of the random initialisation; however, multiple runs of smaller populations showed that, on the order of hundreds of generations, this is not the case. During the course of these experiments, the time performance of the system was evaluated. It was found that the overhead of submitting a Spalloc job merited redesigning the framework to allow multiple models to be evaluated in one job. As a reminder, Spalloc is the current SpiNNaker job submission system which allocates a subset of the entire machine to individual users. This work demonstrated that it is possible and feasible to use a GA to tune the parameters of an SNN model on SpiNNaker.

7.6.6 Future Work

There are a number of avenues for future work in the field of SpiNNaker and optimisation algorithms. Future work could involve the use of different optimisation algorithms, and the application of optimisation algorithms to the conversion of

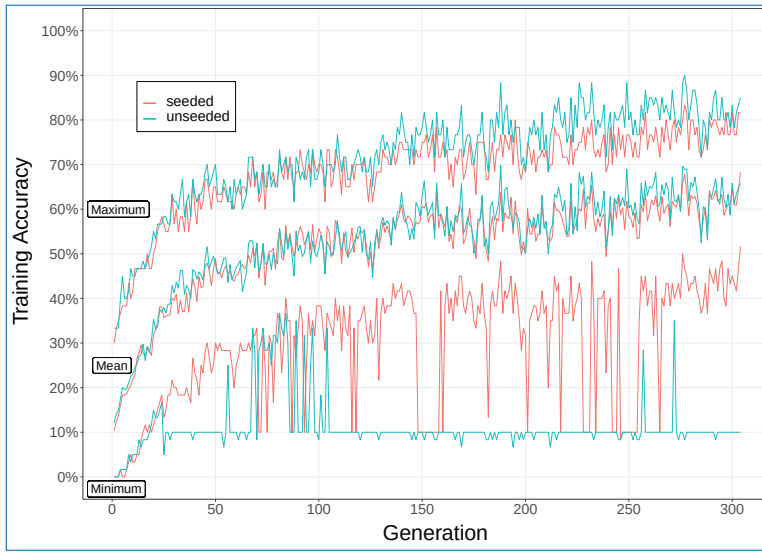

Figure 7.39. A graph comparing number of generations compared with training accuracy for a GA with two populations of 24,000 individuals, each being a SNN model. The seeded population was initialised with 12,000 centre-surround filters, 6,000 positive and 6,000 negative and 12,000 individuals with randomly initialised filters. The unseeded population was initialised with 24,000 random individuals.

ANN models and to uncover novel learning mechanisms in SNNs. Looking more broadly, the development of robust model optimisation frameworks could well lead to a change in the way that research is carried out.

Different EAs

The modification of genes in EAs is random and mutations are undirected; an Evolutionary Strategies (ES) algorithm [15] may well be able to achieve similar results with far fewer evaluations. It does this by mutating the best agent in a population multiple times to create a new generation. The mutation vector of all individuals is then scaled by their performance and totalled to give an approximation of the gradient allowing the next-generation mutation to be in the direction in the solution space which produced the best performance.

One of the key features of SpiNNaker is its asynchronicity, a feature that lends itself to lesser-studied asynchronous steady-state EAs [2]. In these types of models, fitness values are not gathered at the end of a generation as they would be in a typical generational GA, rather models compete between themselves locally. This kind of optimisation algorithm could be run directly on SpiNNaker and would minimise the overheads associated with scattering and gathering data to across multiple chips

and boards. We estimate that running the EA 'on-machine' could half the time taken to evaluate models similar in scale to that of the MNIST test network above.

Machine Learning

A key area of interest is understanding the relationship between SNN and ANN models and translation between them. Deep ANN models trained by error backpropagation give state-of-the-art performance in many benchmark machine learning tasks. An automated optimisation framework could be used to help convert ANN models to SNNs to be run on SpiNNaker. This would dramatically increase energy efficiency for practical applications as well as adding to our understanding of how information is processed in neural networks more generally.

Learning-to-Learn

A current fruitful area of research, dubbed Learning-to-Learn (L2L), applies optimisation techniques to fine tune the hyperparameters of another optimisation algorithm [12]. An example of this could be using a genetic algorithm to evolve the parameters which control how backpropagation performs. This can help find certain parameters and starting conditions conducive of fast learning on novel tasks. With both optimisers working at different time scales, it simulates the slow evolutionary process of genetic variation with the fast time scale acting in a similar way to learned behaviour during a lifetime.

Impact on Computational Neuroscience

By allowing work with biological models that do not have full parameter sets, the focus of researchers working with experimentally derived models could move towards higher levels of abstraction and larger-scale models. This step is an important one to bridge the gap between understanding information processing in the brain and the application of such knowledge in future neuromorphic systems.