Neural Networks 32 (2012) 3–14

Contents lists available at SciVerse ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2012 Special Issue

A forecast-based STDP rule suitable for neuromorphic implementation

S. Davies *, F. Galluppi, A.D. Rast, S.B. Furber

School of Computer Science, The University of Manchester, Oxford Road, Manchester, M13 9PL, United Kingdom

ARTICLE INFO

Keywords:
STDP
Time To Spike
Forecast
Unsupervised learning
SpiNNaker
Izhikevich
Spiking
Neuromorphic

ABSTRACT

Artificial neural networks increasingly involve spiking dynamics to permit greater computational efficiency. This becomes especially attractive for on-chip implementation using dedicated neuromorphic hardware. However, both spiking neural networks and neuromorphic hardware have historically found difficulties in implementing efficient, effective learning rules. The best-known spiking neural network learning paradigm is Spike Timing Dependent Plasticity (STDP) which adjusts the strength of a connection in response to the time difference between the pre- and post-synaptic spikes. Approaches that relate learning features to the membrane potential of the post-synaptic neuron have emerged as possible alternatives to the more common STDP rule, with various implementations and approximations. Here we use a new type of neuromorphic hardware, SpiNNaker, which represents the flexible "neuromimetic" architecture, to demonstrate a new approach to this problem. Based on the standard STDP algorithm with modifications and approximations, a new rule, called STDP TTS (Time-To-Spike) relates the membrane potential with the Long Term Potentiation (LTP) part of the basic STDP rule. Meanwhile, we use the standard STDP rule for the Long Term Depression (LTD) part of the algorithm. We show that on the basis of the membrane potential it is possible to make a statistical prediction of the time needed by the neuron to reach the threshold, and therefore the LTP part of the STDP algorithm can be triggered when the neuron receives a spike. In our system these approximations allow efficient memory access, reducing the overall computational time and the memory bandwidth required. The improvements here presented are significant for real-time applications such as the ones for which the SpiNNaker system has been designed. We present simulation results that show the efficacy of this algorithm using one or more input patterns repeated over the whole time of the simulation. On-chip results show that the STDP TTS algorithm allows the neural network to adapt and detect the incoming pattern with improvements both in the reliability of, and the time required for, consistent output. Through the approximations we suggest in this paper, we introduce a learning rule that is easy to implement both in event-driven simulators and in dedicated hardware, reducing computational complexity relative to the standard STDP rule. Such a rule offers a promising solution, complementary to standard STDP evaluation algorithms, for real-time learning using spiking neural networks in time-critical applications.

1. Introduction

1.1. Learning in biological neural networks

Biological neural networks are known for their ability to learn. Neuroscience research has shown that the learning abilities of a neural network are connected with the plasticity of the neural network itself (Cajal, 1894). Hebb (1949) proposed the first description of a mechanism behind the innate capability to learn in 1949, frequently summarized as: "Cells that fire together, wire together". This general rule emphasises the feature of causality in the activation of neurons: if the pre-synaptic neuron fires an action potential, and as a consequence of this event, the post-synaptic neuron also produces an action potential, then there is a causal relationship between the two events, and therefore the connection (synapse) between the two neurons should be potentiated. However, Hebb's basic postulate takes into account only the *increase* in synaptic efficacy, disregarding the possibility of a *decrease* in synaptic efficacy. Furthermore, this possibility has been analysed in theoretical work and found to be important to avoid saturation of the synaptic efficacy (Bienenstock, Cooper, & Munro, 1982; Sejnowski, 1977). Likewise it is critical if the potential fidelity of the "recall" in associative memory is to be maximized (Willshaw & Dayan, 1990).

Two different processes can account for long-term synaptic depression: the first and most well-known is related to the relative timing of the pre- and post-synaptic spikes (STDP). The second process is called "homeostasis" and happens following a synaptic strengthening event: after synaptic potentiation, all

* Corresponding author.
E-mail addresses: daviess@cs.man.ac.uk (S. Davies), steve.furber@manchester.ac.uk (S.B. Furber).

0893-6080/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2012.02.018

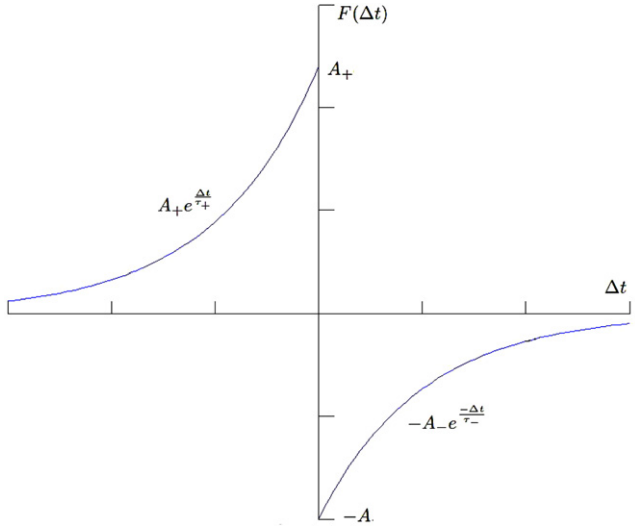


Fig. 1. STDP curve. The horizontal axis represents the time between pre- and post-synaptic spikes ($\Delta t = t_{\text{pre}} - t_{\text{post}}$). The vertical axis represents the synaptic weight modification.

plastic synapses afferent on a neuron should be re-normalized so that the total amount of input to a neuron remains constant (Malsburg, 1973; Sejnowski, 1977). Such renormalisation is easy to achieve in a model, where each node has global visibility of its input weights, but more complex to justify in biology, where since each synapse only has local visibility, there must either be some neuromodulatory intermediary to act as a local proxy, or specific structure in the dynamics that conveys global information through local input spikes. Although these processes imply that the information is stored in the timing of the spike event, Roberts and Bell (2002) highlight a point of contention in neuroscience: it is not yet clear whether information is encoded in the precise timing of the events or in their rate.

The underlying biological details of STDP have been well-studied. In 1996 Markram and Tsodyks (1996) showed that synaptic weight modification occurs when pre- and post-synaptic spikes coincide at low frequencies. More results were described by Bi and Poo (1998) after tests run on hippocampal cells: the results published show a relation between the synaptic weight modification and the relative timings of pre- and post-synaptic spikes. The curve in Fig. 1 shows a functional approximation to the fitted correlation (which is called the STDP curve). These experiments, however, relate only *pairs* of pre- and post-synaptic spikes; it is not yet clear how multiple sets of pre- and post-synaptic spikes influence synaptic weights. Furthermore, the efficacy of the synaptic weight modification has since been shown to be limited by an efficacy time window, and the precise values of the curve vary depending on the system and the neuron type (Abbott & Nelson, 2000; Bi & Poo, 2001; Song & Abbott, 2001).

Biological neural networks involve an important tradeoff between two competing processes (Turrigiano, 1999): the first is the need for change, the second the need for stability. Carpenter and Grossberg (1987) identify this clearly in an important early example of (non-spiking) winner-take-all networks as the "stability–plasticity tradeoff". The first process allows changes that modify the activity pattern in neural networks, creating progressively more emphasized differences between neurons, but with the effect of destabilizing it (Kube, Herzog, Michaelis, de Lima, & Voigt, 2008). The second process stabilizes the activity in the neural network, making the activity uniform. In particular, the first process is associated with learning novel inputs, while the second normalizes the input of a neuron on the basis of its mean firing rate, a state termed "homeostatic equilibrium". Gilson, Burkitt, Grayden, Thomas, and van Hemmen (2009) make an extensive analysis of this tradeoff for a general neural network and find that the network structure is critical: in order to obtain stable networks that can successfully learn differentiated patterns, the inputs must be arranged in correlated input groupings, while the initial distribution of weights on the input neurons must be nonuniform. Grossberg and Versace (2008) take a more explicit approach, significantly extending and transforming the ART (Adaptive Resonance Theory) model into a spiking representation incorporating the same tradeoffs. As per earlier ART models this uses an auxiliary resetting network to detect novel inputs and make category groups eligible for learning. Both of these approaches emphasize the need for nonuniform initial network structure.

At a network level, Hebb's general postulate has led to a variety of "Hebbian" learning rules. While initially the primary attraction of Hebbian rules was their ability to support unsupervised learning, the discovery of STDP in biology – and its subsequent identification as a Hebbian process (Gerstner & Kistler, 2002) – provide new motivation by suggesting or explaining biological learning mechanisms. Various different learning rules have been examined theoretically and in simulation in an attempt to find biologically realistic models. The field of neural coding provides several important examples of the use of STDP in practice. The timing sensitivity of STDP provides an obvious mechanism for network synchronisation. Izhikevich (2006) has proposed several synchronous networks based upon STDP that use it to tune neurons with convergent delays in order to form timing-locked logical groups. However, while synchronous networks offer a *dynamic* explanation of biological patterns, they have not thus far provided a complete *functional* explanation - that is, the role of synchronisation in the processing remains disputed (Masuda & Kori, 2007). Winner-take-all (WTA) networks provide explicit functional coding, either for the formation of stimulus-selective cortical columns (Song & Abbott, 2001), or for methods of pattern selection (Masquelier, Guyonneau, & Thorpe, 2008a). Rank Order Coding (Thorpe & Gautrais, 1998) may be considered as a form of time dependent WTA neural coding, where information is encoded in the order in which a neuron's inputs fire. STDP has been used in conjunction of Rank Order Coding in the emergence of selective orientation receptive fields (Delorme, 2001) and in the unsupervised learning of visual features (Masquelier & Thorpe, 2007). Reinforcement learning models prescribe even more explicit functional roles, such as the use of STDP to construct a predictor (Rao & Sejnowski, 2001). Critically, STDP remains to date the only biologically plausible underlying mechanism to demonstrate functional learning rules for spiking neural networks (Arena, Fortuna, Frasca, Patané, & Sala, 2007), and as such forms a central part of spiking models.

1.2. Algorithmic techniques for synaptic plasticity

When modelling neural networks in simulation, it is necessary to determine a precise form for the learning rule. Multiple learning algorithms have emerged, attempting various tradeoffs between computational complexity, biological realism, and analytical tractability. The most well-known STDP rule considers every possible combination of pre- and post-synaptic spikes, modifying correspondingly the synaptic weight (Song & Abbott, 2001). This rule has been used in several tests (e.g. Guyonneau, VanRullen, & Thorpe, 2005) and has proven to replicate the biology with some degree of accuracy. Burkitt, Gilson, and van Hemmen (2007) have given a mathematical and statistical characterization of this rule. One particular property of the STDP emerges from the tests of Guyonneau et al. (2005): a neuron which identifies an input pattern repeatedly advances the timing of identification of this

pattern earlier and earlier, until it reaches the earliest group of spikes that begins the pattern.

An approximation of this algorithm is nearest-neighbour (or spike-pair) STDP. This is the simplest learning rule that has shown some biological realism in experiments (Masquelier et al., 2008a; Masquelier, Guyonneau, & Thorpe, 2008b). It is known to approximate at least one biologically relevant feature suggested by Izhikevich and Desai (2003): post-synaptic spikes propagate back into the dendritic spines, resetting them. As a result, the latest post-synaptic spike "erases" the effect of earlier pre-synaptic spikes. This rule is frequently implemented in analogue hardware neural network chips (e.g. Indiveri, Chicca, & Douglas, 2006) since it is simple: the timing of only one pre- and post-synaptic spike is needed to trigger synaptic potentiation and depression, eliminating the need for complex, area-intensive spike buffering.

Bienenstock et al. (1982) have created a theoretical learning rule (called the BCM rule after the authors' initials): the basis of this algorithm is that the instantaneous firing *rate* rather than individual spikes (as for STDP) set the pattern of weight modifications. Synaptic inputs that drive the post-synaptic neuron to high firing rates are potentiated, while inputs that produce low firing rates are depressed. Izhikevich and Desai (2003) have proven an equivalence of the BCM rule and the STDP spike-pair rule, opening both to common theoretical treatment and adding flexibility in specific model implementation choices.

More variants of the STDP rule have been presented which take into account trains of three and four pre- and post-synaptic spikes (Pfister & Gerstner, 2006; Wang, Gerkin, Nauen, & Bi, 2005). These rules try to explain a biological effect that shows synaptic depression when neurons are subject to high input firing rates, where standard STDP would potentiate those synapses (Sjöström & Gerstner, 2010). In particular, in Pfister and Gerstner (2006) a novel description of synaptic plasticity based on triplets of spikes seems to reproduce neurobiological experimental results with more accuracy, particularly under conditions where the standard STDP rule shows limitations. The described behaviour seems to match the BCM rule described before (Pfister & Gerstner, 2006), even if the mathematical proof is limited to some statistical constraints on the input.

The algorithms that directly relate spike timings to synaptic weight modification have been shown more or less to be biologically plausible. Recently, a new type of algorithm based on the use of other observables or model variables as proxies for the spike timing has started to emerge. Such algorithms dramatically reduce the computational load since the model does not have to compute each independent spike time. One new class of those, in particular, relate the synaptic weight modification to the (pre- or post-synaptic) neuron membrane potential (or by a low-pass filtered version of it) (e.g. Clopath, Busing, Vasilaki, & Gerstner, 2010; MacNeil & Eliasmith, 2011). Artola, Brocher, and Singer (1990) describe the biological principles supporting this class of algorithm: depending on whether the membrane potential of a post-synaptic neuron is above or below an LTP or LTD threshold, the appropriate update is triggered. This assumes a form of voltage gating of the synaptic update, much like the voltage gating term for transmission of NMDA synapses.

In this context a new algorithm has been proposed by Davies, Rast, Galluppi, and Furber (2011) that relates the synaptic weight modification with the post-synaptic membrane potential in a network of Izhikevich neurons. The paper presents an implementation of the STDP TTS (Time-To-Spike) in a Matlab environment, and it shows that the new algorithm is able to detect one or multiple patterns in a noisy background, lock the identification and tune to the earliest possible group of spikes. In this paper we present an implementation of the STDP TTS learning rule on the SpiNNaker neuromimetic simulator and compare it with the best-known STDP rule to show that it is less expensive from both a computational point of view and a memory utilisation point of view, while being able to detect patterns with performance comparable to the standard STDP rule.

2. Spiking neural network simulation hardware

Here we first analyse and classify some of the hardware used for simulation of spiking neural networks in the research field, with a particular focus on the learning features, if available (Brette et al., 2007). Then we introduce the SpiNNaker system as a general-purpose hardware spiking neural network simulator. We will use SpiNNaker to perform the tests on the new STDP TTS learning rule.

- *BrainScaleS* (previously FACETS): This project has developed an analogue hardware neural network simulator, implementing adaptive exponential leaky integrate-and-fire neurons with learning capabilities (Schemmel, Brüderle, Meier, & Ostendorf, 2007), running 10^4 times faster than real time. The learning algorithm implemented is the spike-pair STDP rule, with the possibility of additive or multiplicative variation of the synaptic weight.
- *VLSI chips for artificial neurons*: This project developed a VLSI chip that is able to simulate adaptive exponential leaky integrate-and-fire neurons running in real time (Livi & Indiveri, 2009). Synapses are plastic (Indiveri, Stefanini, & Chicca, 2010).
- *Neurogrid*: This project built an analogue hardware simulator with learning capabilities (Arthur & Boahen, 2006).
- *Blue Brain*: This project aims to use the computational power of the BlueGene/L supercomputer to simulate neural networks (Migliore, Cannia, Lytton, Markram, & Hines, 2006). This is a software simulator, therefore it is possible to program the neuron types and the learning models required. However, this simulator does not run in real time.

2.1. SpiNNaker

SpiNNaker is a hardware-based real-time universal neural network simulator using an event-driven model of computation. This project involves the design of a chip and the development of dedicated software to simulate neural networks. This system tries to mimic in various ways the features of biological neural networks:

- *Native parallelism*: in biological neural networks, each neuron is a primitive computational element within a massively parallel computational network. Likewise SpiNNaker uses parallel computation in a symmetric multicore system;
- *Event-driven behaviour*: in biology, neurons communicate through spikes. The SpiNNaker architecture uses AER packets to transmit neural signals between processors (and therefore neurons) over a configurable packet switched network with asynchronous interconnection;
- *Distributed incoherent memory*: in biology, no central global memory exists. Neurons use only local information to process incoming stimuli. In the SpiNNaker architecture we use a distributed hierarchy of memories: a Tightly-Coupled Memory (TCM) local to to each of the cores and an SDRAM local to each chip partitioned between on-chip cores, with an incoherent access mechanism;
- *Reconfigurability*: in biology, neurons are plastic. This means that the interconnections change both in shape and in strength, while the neural network evolves. Likewise in the SpiNNaker architecture the routing system allows for reconfiguration on-the-fly.

Here we introduce the main features of the SpiNNaker system. Complete details of the system and of its performance can be found in Furber, Temple, and Brown (2006), Plana et al. (2007), Rast et al. (2010), Rast et al. (2011) and Rast et al. (2011).

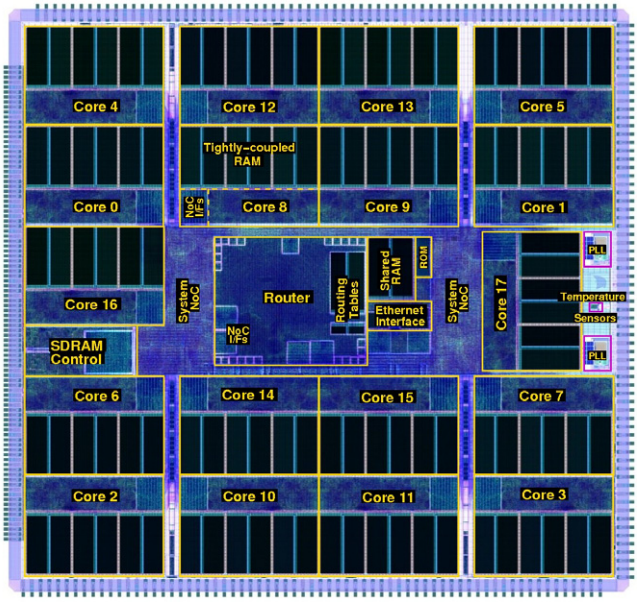


Fig. 2. SpiNNaker chip layout with labels for functional blocks. Image taken from the GDS2 plot sent to manufacture. The chip was fabricated at UMC using their 130e-llsp low-power process. Die size is 10 × 10 mm.

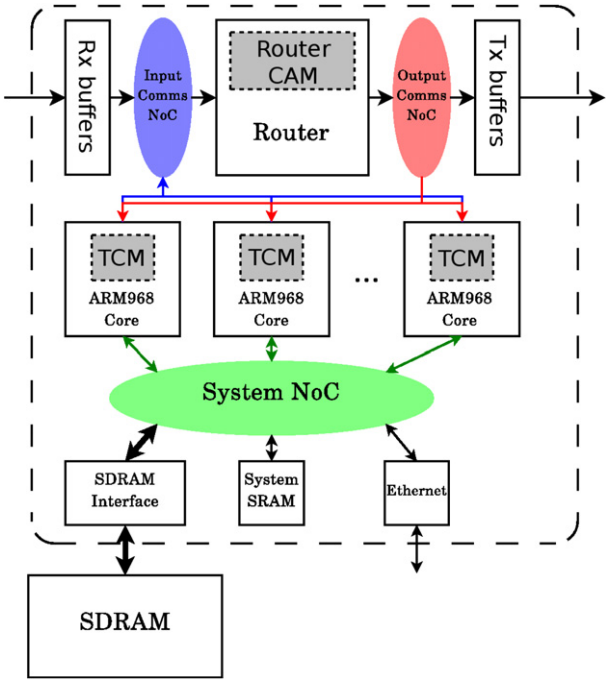


Fig. 3. SpiNNaker chip block diagram.

2.1.1. Hardware

SpiNNaker integrates the essential elements of the neuromimetic architecture: a hardware model designed to support flexibility in model exploration while implementing as many known features of the neural model of computation as practicable explicitly in hardware for maximal performance (Furber et al., 2006; Rast et al., 2011, 2010). The core of the SpiNNaker system is the SpiNNaker node: a multiprocessor chip including 18 ARM968 processors (see Fig. 2), each running at 200 MHz. Each core has a 96 kB TCM (Tightly Coupled Memory) containing local instructions and data. In addition, a 128 MB SDRAM memory is available to all the cores of a chip. Typically the system configuration will generate an address map that partitions this SDRAM into exclusive areas for each core. A network router (Plana et al., 2007) provides interconnection features to the cores internally and externally to the chip (see Fig. 3). While the ARM cores are off-the-shelf general purpose programmable processors, the router is a completely custom component optimized for spiking models (although its use is not limited to spiking neural network simulation).

SpiNNaker chips integrate into a large-scale system normally consisting of a regular two-dimensional hexagonal array of up to 256 × 256 chips (see Fig. 5). The whole network is then wrapped to form a toroidal shape (Fig. 4). Some of the nodes in this network are connected to the external world by an Ethernet link, through which it is possible to send data, code, and commands to the SpiNNaker system. It is worth noting in particular the programmability of this system: although it has been designed to perform with spiking neural network simulations, the general purpose cores embedded in the SpiNNaker chip make this hardware versatile and usable also in other contexts and for different purposes.

2.1.2. Software

The software developed for this system has been designed to perform with maximal power efficiency. The processor is kept in low-power "sleep" mode with interrupts enabled, as much as possible. When an interrupt is received, the processor performs the required actions to respond to the interrupt requested and then returns to sleep. There are three main interrupt sources involved in the simulation of neural networks:

1. *Timer interrupt*: this interrupt is triggered to integrate the neural equation(s) over a time-step. Currently, for biological realism, we use a time step of one millisecond, but this is configurable through the software to comply with the neural model in use and/or the simulation requirements. This is the main interrupt that advances the simulation.
2. *Packet received interrupt*: this interrupt is triggered for every SpiNNaker AER packet (representing a neural action potential) that is received by the destination core. The received packet must be transferred as quickly as possible into an internal software buffer to avoid the possibility of congestion of the network router, and consequently packet loss from drops by the router itself. The queued packet will then trigger a DMA operation that brings inside the core the synaptic data structures required to compute the input to the destination neurons.
3. *DMA transfer complete interrupt*: this interrupt signals the completion of the transfer of synaptic data structures (held off-chip in the SDRAM memory) into the Data TCM of the core. This allows the required learning algorithm to be triggered, and subsequently the transmission of the spike to the destination neurons.

To transmit the stimuli to the neurons we use a software buffer (with programmable length) that simulates the synaptic delay at the destination core. In our implementation this is a 16-slot circular buffer where each slot is a software-configurable time step. As configured we can allow a maximum delay of 16 ms. These slots are loaded with the injected current during the DMA transfer complete interrupt, and the value is reset once the input has been injected into the destination neuron.

The three interrupts described above operate at different priorities so that the packet received interrupt has the maximum priority, DMA transfer completes the next level of priority, and timer interrupt, which requires the longest time to complete (since it needs to update all the neurons in a core), the lowest.

2.1.3. Learning rule

The current learning rule developed for the SpiNNaker simulator is based on the standard STDP algorithm. Usually in software

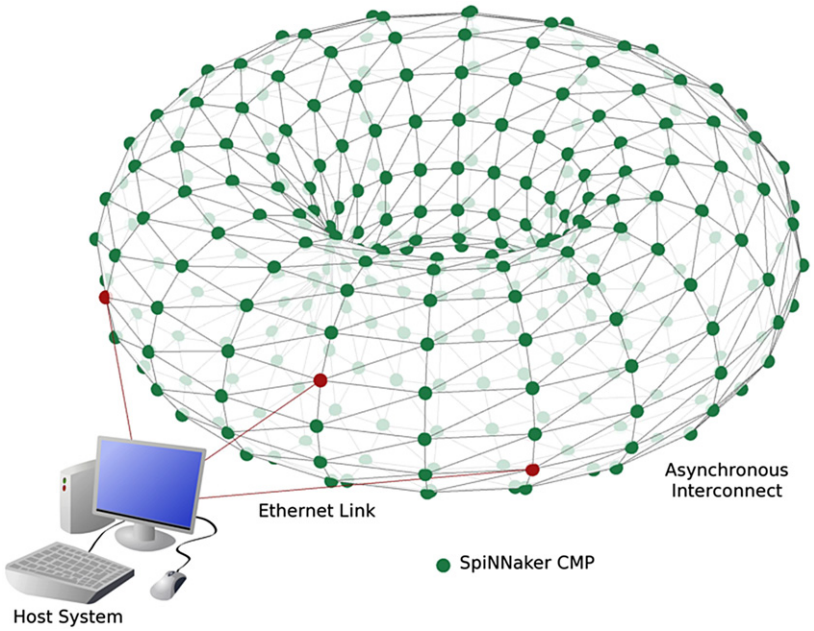


Fig. 4. View of the complete SpiNNaker system: the two-dimensional array of chips is wrapped in a toroidally shaped network.

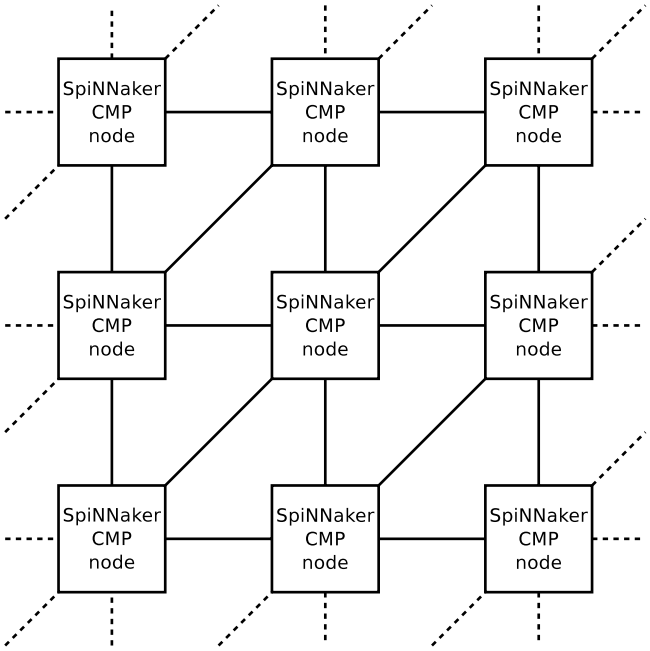


Fig. 5. Two-dimensional array of SpiNNaker chips with details of the interconnection.

simulators this rule is implemented so that when a spike is received by the post-synaptic neuron, LTD is triggered with respect to all the spikes that the destination neuron has emitted in the relevant time window. On the other side, when an action potential is emitted by a neuron, LTP is triggered with respect to all the incoming spikes from all the synapses (see Fig. 6(a)).

In the SpiNNaker system this implementation is not easily programmable: the synaptic data structures are available to process only when a spike is received. Therefore a novel model has been implemented (Jin, Rast, Galluppi, Davies, & Furber, 2010; Jin, Rast, Galluppi, Khan, & Furber, 2009) to store all the information related to the incoming and outgoing spikes and trigger LTD and LTP when all the relevant information is available locally in the core (see Fig. 6(b)). This execution model has been called the Deferred-Event Model (Rast, Jin, Khan, & Furber, 2009).

This type of implementation has an impact on the memory occupied by the data stored for the records (incoming spike record and outgoing spike record), and on the computation required to perform the synaptic weight update. These impacts are described in the cited articles.

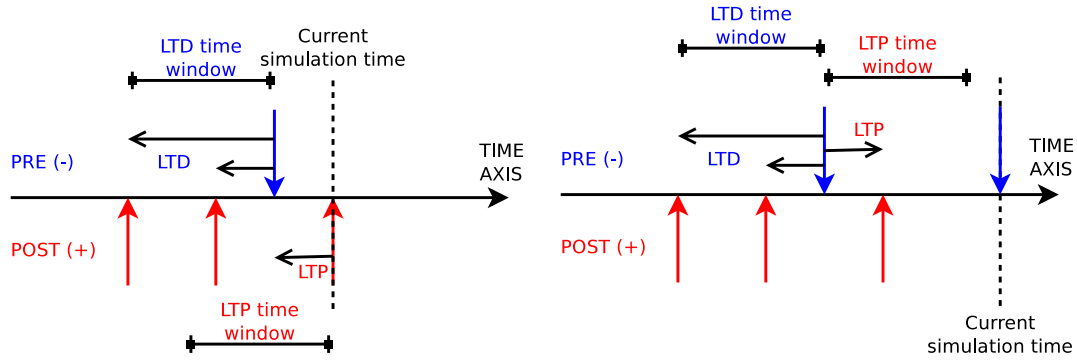
3. The STDP TTS learning rule

In this paper we present a novel learning algorithm for spiking neural networks. The key feature of this new learning algorithm is that it is able to update the synaptic weight as soon as a pre-synaptic action potential is received by the post-synaptic neuron. To achieve this, we trigger LTD using the standard STDP rule, while for LTP we trigger the synaptic modification in relation to the membrane potential value of the post-synaptic neuron.

To perform this algorithm a relation between the membrane potential and the estimated time to spike of a neuron is needed. To determine this we use a statistical approach: the basic idea is that the higher the membrane potential is, the sooner the neuron is going to fire (with some probability). This forecast model is called "Time-To-Spike" (TTS), and the LTP part is similar to the spike-pair STDP rule, although the time of the outgoing spike is unknown at the moment of the weight modification, and is estimated on a statistical basis.

In contrast to the standard STDP model implemented in the SpiNNaker system, which requires the Deferred Event Model as described above (Jin et al., 2010, 2009), this algorithm computes both potentiation and depression (LTP and LTD) when a spike arrives at its destination.

Characterization of this learning rule goes through three steps: the first is a statistical analysis of the function that forecasts the "Time-To-Spike" on the basis of the post-synaptic neuron membrane potential. To do so we extract the statistical features required using a Matlab neural network simulator composed of Izhikevich neurons (Izhikevich, 2006). The second step is to validate this learning rule using the Matlab simulator described before, using a neural network designed to identify patterns present on the input neuron layer. The third step implements a further approximation to this rule on the SpiNNaker system to



(a) Standard STDP software implementation.

(b) STDP with Deferred Event Driven Model.

Fig. 6. Implementation of the standard STDP learning rule.

check whether it is possible to use a spike-pair STDP learning rule, as opposed to the standard STDP rule implemented in the Matlab simulator.

4. Extraction of the statistical features

For the purpose of this paper we focus on Izhikevich neurons described by two ordinary differential equations (Izhikevich, 2003):

$$\begin{cases} \dot{v} = 0.04v^2 + 5v + 140 - u - I \\ \dot{u} = a(b \cdot v - u) \end{cases} \quad (1)$$

$$\text{if } v \geq 30 \text{ mV then } v = c, u = u + d. \quad (2)$$

Using a network composed of 4000 Izhikevich neurons randomly interconnected with random delays we estimate the relation between the membrane potential and the time required for this neuron to fire. Two types of neurons are used in this network:

1. Tonic spiking neurons: $(a = 0.02; b = 0.2; c = -65; d = 8)$;
2. Fast spiking neurons: $(a = 0.1; b = 0.2; c = -65; d = 2)$.

There are 3200 neurons of the first class each connected to 25 neurons of both classes with excitatory synapses. The synaptic strength of these connections is 10 nA/μF. In addition there are other 800 neurons of the second class each connected to 25 neurons randomly chosen between the tonic spiking neurons with inhibitory synapses. The strength of the connection is −5 nA/μF.

The whole network is stimulated by the injection of a constant current of 20 nA/μF into 60 randomly chosen neurons in the first class and 20 randomly chosen neurons in the second class. No synaptic plasticity is enabled for this test. The simulation is run for 1000 steps of 1 ms each, for an overall simulation time of 1 s.

During the simulation the software records the membrane potential for all the neurons in both populations and the incoming spike times. At the end of the simulation the membrane potential record is post-processed: for every spike received by a neuron a couple (membrane potential, Time-To-Spike) is computed (see Fig. 7). Finally the couples computed before are grouped and sorted on the basis of the membrane potential, and a mean value of all the elements in a group is computed.

The difference between the two parts of the graph in Fig. 8(a) can be explained by examining the Izhikevich neuron phase space (Fig. 9). If the value of the neuron state variables v and u are in the half-plane on the left of the firing threshold (the dashed line and the BC part of the parabola in the graph), the neuron state will move towards equilibrium point A in the absence of input. By contrast, if the neuron state is in the right half-plane, the neuron state will move towards the firing condition ($v_{\text{membrane}} \geq 30$ mV). The threshold that identifies the two half-planes corresponds to the discontinuity between the two sections of the graph in Fig. 8(a).

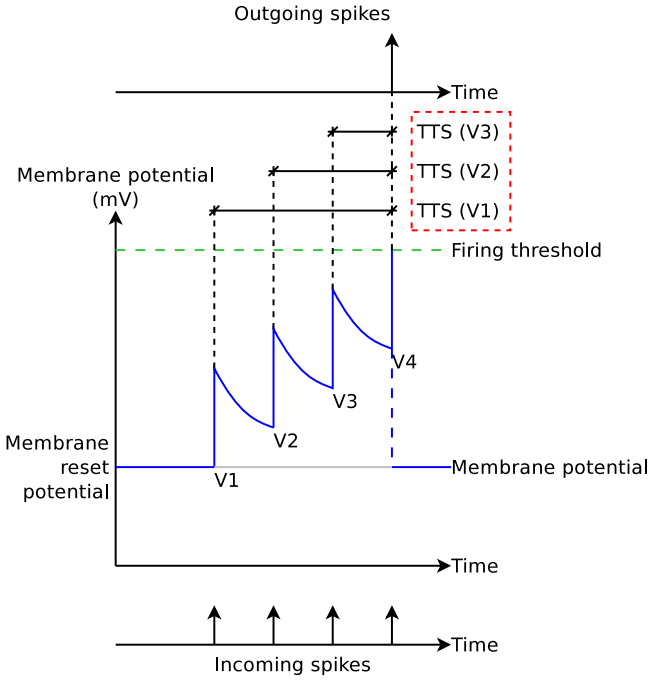
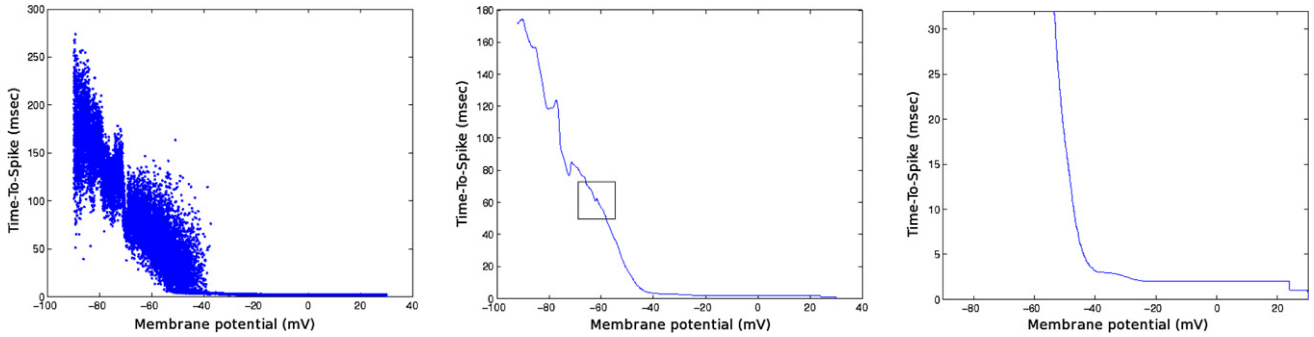


Fig. 7. Example of computation of the Time-To-Spike (TTS) of a neuron.

4.1. Post-processing of the statistical function

To remove some noise from the graph in Fig. 8(a), we applied a sliding window filter over an interval of 1024 equally-spaced membrane potential values, so that variation of the Time-To-Spike in the membrane potential/time-to-spike function is smoother. The outcome of this post-processing is shown in Fig. 8(b). While the section on the right side of the graph is linear as described before, the section on the left shows some fluctuations. In particular, around the point (−60 mV; 60 ms) (see detailed Fig. 8(b)) it is possible to note the first evident nonmonotonicity: in a short interval around those values, for lower values of the membrane potential the neuron is predicted to fire sooner. We interpret this as a statistical aberration, and the more we move towards the left part of the graph, the more evident these aberrations become. To avoid encountering such regions we focus on a window of 32 ms (Fig. 8(c)).

The figure shows a function characterized by two segments: the first passes through the points (30 mV; 0 ms) and (−40 mV; 3 ms). The second passes through the point (−40 mV; 3 ms). We use the second edge of the segment as a parameter for the forecast (parameter "L"). Fig. 10 then summarises the relation function between the membrane potential and the estimated time to spike.



(a) Raw version of the relation function. The graph can be divided into two sections with very different behaviours, the discontinuity occurring at about −40 mV. For membrane potentials above this value, estimated time to spike is linear between 3 ms and 0 ms. For values below −40 mV the graph is very noisy.

(b) Filtered version of the relation function. The filter applied is a sliding window over an interval of 1024 equally-spaced membrane potential values. The square highlights the first incongruence in the forecast of the Time-To-Spike: in this region the lower the membrane potential, the sooner the neuron fires.

(c) Enlargement of the 32 ms time window used for the forecast function.

Fig. 8. Function that relates the membrane potential in mV (on the horizontal axis) and the estimated time to spike in ms (on the vertical axis).

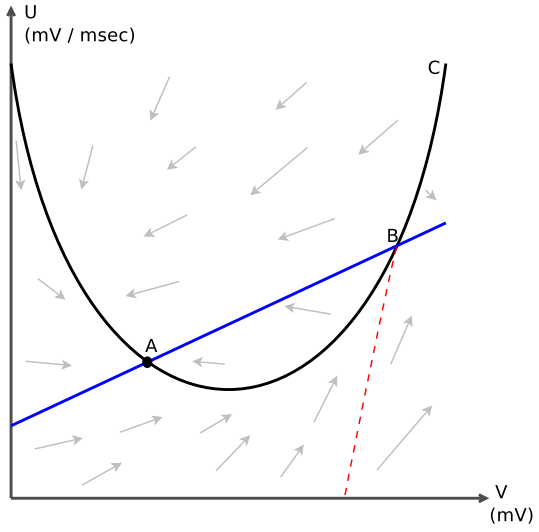


Fig. 9. Izhikevich neuron state phase plane. The horizontal axis is the membrane potential variable v. The vertical axis is the membrane recovery variable u.

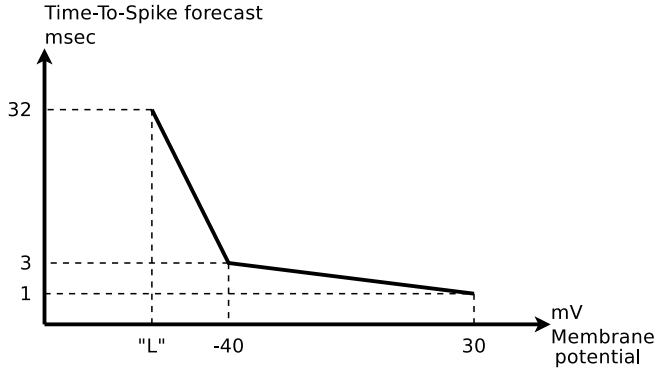


Fig. 10. Relation between membrane potential and time to spike of the neuron.

5. Simulation and testing in Matlab

The first step in the simulation of this new algorithm has been to implement it as an extension of the Matlab simulator provided by Izhikevich (2006). The neural network used for the simulations has two layers of neurons: the input layer is composed of 100 neurons connected to an output layer of one, two or four neurons (see Fig. 12). The synapses between the two layers connect all the neurons of the input layer to all the neurons of the output layer.

All the neurons used are tonic spiking Izhikevich neurons. In the four output neuron simulations the output neurons inhibit each other for 5 ms with a synaptic strength that decreases linearly from −5 nA/μF to −1 nA/μF. Here we introduce two sets of experiments: the first uses the standard STDP algorithm, while the second set uses the new STDP TTS learning rule. The STDP curve used during these experiments is defined by the function:

$$F(\Delta t) = \begin{cases} A_+ e^{\frac{\Delta t}{\tau_+}} & \Delta t < 0 \\ -A_- e^{\frac{-\Delta t}{\tau_-}} & \Delta t \geq 0. \end{cases} \quad (3)$$

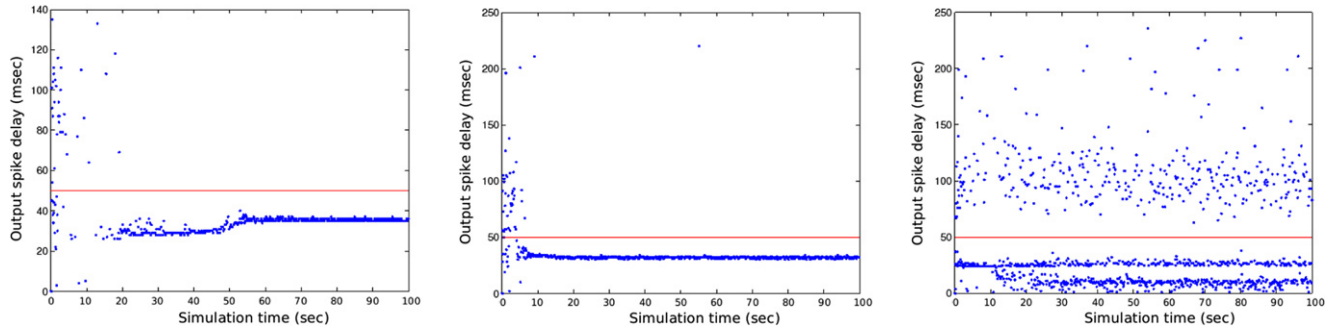
The parameters we used are (Jin et al., 2010, 2009; Song, Miller, & Abbott, 2000): $\tau_+ = \tau_- = 20$ ms, $A_+ = 0.1$ and $A_- = 0.12$, LTD time window = LTP time window = 32 ms.

An example of the input provided to the network is shown in Fig. 13: the input layer is divided in two subclasses partitioned equally. The first subclass receives only noise while the second class receives one (or two) patterns hidden in the noise. When there are two input patterns, they are interleaved between each other (1-2-1-2-1-...). The input noise is generated through a Poisson process with a mean firing rate of 50 Hz. The pattern to be identified is generated in the same way as the noise, and has the same firing rate as the noise, therefore it is not easily identifiable from an inspection of the raster plot of the input (see Fig. 13). The inter-pattern time is itself a Poisson process with a mean modulation frequency of 9 Hz. However patterns cannot be present across the boundaries of a second. Therefore this reduces slightly the mean presence of the pattern from the planned 9 Hz.

The outcome of all the tests run throughout this paper are in the form of "scatter plots" (e.g. Fig. 15). In these graphs the horizontal axis represent the simulation time (in seconds). The vertical axis represents the time from the beginning of the pattern injection to the output spike (in milliseconds). Every dot represents an action potential emitted by (one of) the output neuron(s), as described in the figure caption. The spikes emitted within the time window of the input pattern (delimited by the red line in the graphs) are regarded as positive identification. All the spikes that are above the line delimiting the pattern time window are considered false positives and therefore noise in the output.

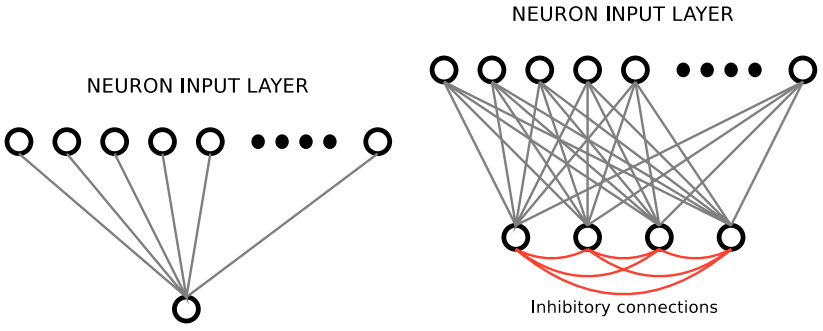
5.1. Results from Matlab simulations

This section presents the results obtained from the Matlab simulator using the standard STDP rule or the plasticity with Time-To-Spike forecast.



(a) $L = -60$ mV. (b) $L = -65$ mV. (c) $L = -70$ mV.

Fig. 11. Scatter plots for STDP with TTS forecast for various values of the L parameter, one output neuron network, one input pattern, simulated in Matlab.



(a) Network structure: 100 input neurons—1 output neuron.

(b) Network structure: 100 input neurons—4 output neurons. The output neurons inhibits each other (red links).

Fig. 12. Structure of the neural networks used in the tests: 100 input neurons and 1 or 4 output neurons. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

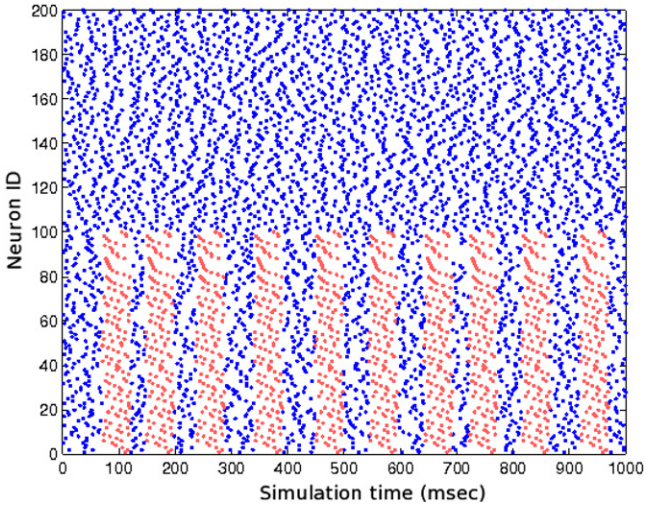


Fig. 13. Example of raster plot of the input pattern for an input layer of 200 neurons and pattern sent to the first half of this population. The input pattern is highlighted in red. In our simulation the input will be generated for 100 input neurons of which only half will receive the input pattern. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5.1.1. Standard STDP results

The first series of tests used a single output neuron with a single input pattern. Parameters for this test are set as follows: initial synaptic weight is 5 nA/μF. Maximum synaptic weight is 10 nA/μF. The output of this test is shown in Fig. 15. At the beginning the neuron requires some time to identify the pattern. Once it has been detected the neuron locks on it and advances the detection earlier and earlier in the pattern. However, as the detection advances earlier and earlier in the input pattern it is possible also to see an increase in the noise generated by the output neuron. In a second series of tests, we used four output neurons with two input patterns. Results from these tests have been discussed in Davies et al. (2011).

5.1.2. Plasticity with Time-To-Spike forecast

We first test the behaviour of this new learning rule with regards to the "L" parameter using a network with only one output neuron and one input pattern injected between the noise. Finally we use a four-output neuron network (Fig. 12(b)) with two input patterns to check how multiple neurons can interact in the learning of multiple patterns, and how it compares with the standard STDP rule.

1. *Single output neuron–Single input pattern*:

   For all these tests the parameters are set as follows: initial synaptic weight is 4 nA/μF. Maximum synaptic weight is 4 nA/μF. Modifications to the L parameter cause different behaviour in learning. Some examples are described in Fig. 11. For $L = -60$ mV the output neuron delays the identification of the input pattern. In some simulations the output neuron delays the identification beyond the boundary of the input pattern and it eventually "dies". For $L = -65$ mV the output neuron keeps the identification time constant, reducing the output noise significantly compared to the standard STDP rule. For $L = -70$ mV the output neuron identifies the input pattern twice for every injection. The output noise, however, is fairly high. For subsequent tests we use $L = -65$ mV for the properties presented here.

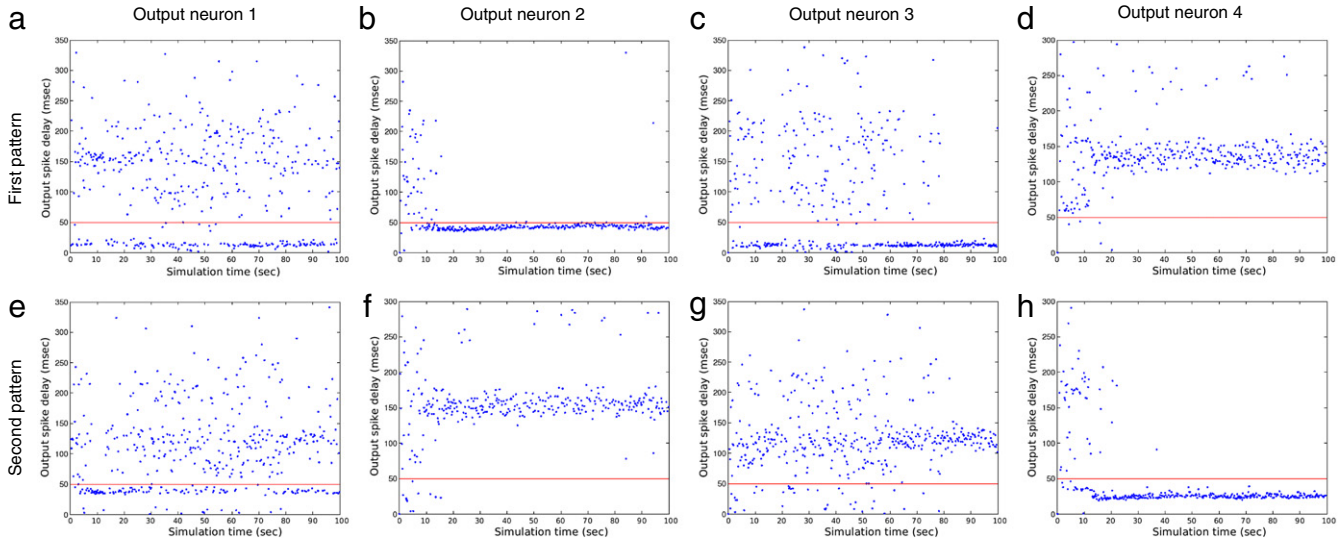


Fig. 14. Scatter plots for STDP with TTS forecast: two input patterns, four output neurons, simulated in Matlab. The learning parameter L is set to $L = -65$ mV. The first row is related to the first input pattern, while the second row is related to the second input pattern. Each of the columns refers to a single output neuron.

2. *Four output neurons—Two input patterns*:

   For this test the parameters are set as follows: initial synaptic weight is uniform in the interval $[0 : 2]$ nA/μF. Maximum synaptic weight is 2 nA/μF. In the results of this simulation (see Fig. 14) is possible to see that the output neurons 2 and 3 identify and lock on the first pattern. Output neuron 4 identifies and locks on the second pattern. Output neuron 1 does not have a clear lock on either of the two input patterns. This may be due to contention in the identification of a specific section of the input patterns given by the winner-take-all configuration of the output neurons.

   As it is possible to see in Fig. 14(d), (f) and (g) there is a "ghost" effect. This is connected with the presence of two different interleaved input patterns whose inter-time is random. When a neuron identifies one of the two input patterns it fires with a delay from the beginning of the identified pattern, and with a greater delay from the beginning of the other pattern which preceded the identified one. Given the randomness of the inter-pattern time, the result is the "ghost" effect described. This effect is noticeable in all the multiple pattern simulations presented in this paper.

6. Simulation and testing on SpiNNaker

The implementation of the standard STDP rule follows the Deferred-Event Model described earlier (Jin et al., 2010, 2009).

The implementation of the STDP TTS rule on the SpiNNaker system has been modified from the implementation in Matlab. While in Matlab the LTD part of the STDP algorithm is triggered considering all relevant pairs of outgoing spikes with the received one, in the SpiNNaker system, for memory and computational efficiency, we compute the LTD part using only the timing of the last outgoing spike with respect to the spike just received, which is closer to the spike-pair STDP rule. Moreover, moving the algorithm from Matlab to the SpiNNaker chip we changed from a floating-point to a fixed-point arithmetic unit. This introduces some numerical differences.

To test this variation the implementation of the algorithm uses an approximation that eases the complexity for a first test: the voltage sample to trigger LTP is taken when the spike is received at the destination core, instead of sampling the membrane potential of the neuron after the synaptic delay. However this approximation has been kept as low as possible using the shortest possible synaptic delay for the incoming spikes. The minimum synaptic delay is 1 ms, and this is believed to be negligible when computing the synaptic potentiation.

6.1. Results from the SpiNNaker system

6.1.1. Plasticity with Time-To-Spike forecast

For these tests we have chosen $L = -65$ mV. To compensate for use of the spike-pair rule instead of the standard STDP rule for synaptic depression (as tested in Matlab), long term depression has been doubled both in strength and in the efficacy time window. The parameters used for this simulation are the following: $\tau_+ = \tau_- = 20$ ms, $A_+ = 0.1$ and $A_- = 0.24$, LTP time window = 32 ms, LTD time window = 64 ms. The initial synaptic weights are set with a uniform distribution in the interval $[4 : 5]$ nA/μF. The maximum synaptic weight is set to 10 nA/μF. The inhibitory connection between neurons is set to use an exponential decay with maximum initial value equal to -20 nA/μF and decay time constant $\tau = 20$ ms.

1. *Single output neuron–Single input pattern*:

   The result of this test is presented in Fig. 17. There are observable differences from the Matlab results shown in Fig. 11(b). These are believed to be connected with the different implementation of the TTS rule between the Matlab implementation and the SpiNNaker implementation, as described before. However it is possible to note that the output neuron emits action potentials (in the majority of cases) within the time window of the input pattern.
2. *Four output neurons–Two input patterns*:

   The results of this test are presented in Fig. 16. Output neuron 1 does not lock to either of the two input patterns because it is pushed away by the inhibition signals of all the other output neurons. Output neuron 2 locks on the second pattern. Output neuron 3 locks on the first pattern. Output neuron 4 locks on both the patterns, with a peculiarity: the portion of the pattern on which this neuron locks is very close (identical in some points) to the section identified by output neuron 2. This will be discussed in the next section.

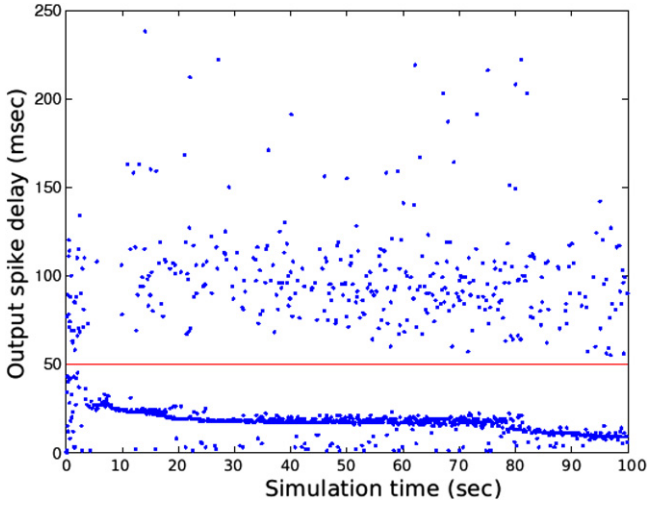


Fig. 15. Scatter plot for standard STDP, one output neuron network, one input pattern, simulated in Matlab.

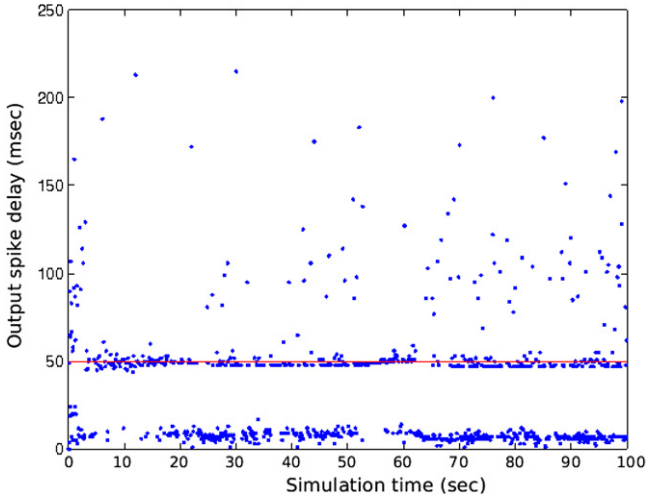


Fig. 17. Scatter plot for STDP with TTS forecast and $L = -65$ mV, one output neuron network, one input pattern, simulated on the SpiNNaker system.

7. Discussion

The TTS algorithm permits aggressive improvement in computational and memory efficiency. However, *network* response may slow because it is not able to tune to the earliest spike groups of the input patterns (Fig. 18). A weak winner-take-all configuration with mutual inhibition in the four-output-neuron simulations is used to discourage two neurons from identifying the same portion of the input pattern. However, in some cases (Fig. 16) the inhibitory connection is not fast enough to prevent two neurons from learning the same pattern. Because the TTS rule *predicts* future spikes, weight changes may occur even if in actual fact the output neuron does not produce an action potential. All that is required to trigger LTP is that the membrane potential reaches the threshold imposed by the L parameter (-65 mV in our tests). Furthermore, this prediction is based on the membrane potential prior to adding input delay corrections. Thus the TTS rule may not account for delayed inhibitory inputs, such as can happen in the WTA configuration. The result can be simultaneous potentiation of input synapses to neurons that are rivals for the same input representation. One solution is to add more randomness in the neuron and connection parameters to break network symmetry.

Small variations of the learning parameter L result in very different behaviour: while some values produce results similar to the standard learning behaviour ($L = -65$ mV, Fig. 11(b)), others have very noisy behaviour ($L = -70$ mv, Fig. 11(c)) or even anti-learning behaviour ($L = -60$ mV, Fig. 11(a)). If we move the L parameter to a value which is too low, the network may adapt to random noise present on the input, which increases the mean membrane potential of the output neuron. In our tests no random noise was added on the input pattern. Indeed, it could modify the forecast function because of its effect upon the mean membrane potential. On the other hand, opening the forecast time window to greater values than what we chose in this paper may lead to unpredictable and/or unstable behaviour.

The algorithm discussed in this paper is based on a statistical approach which may be sensitive to model specifics. We have extracted the statistical behaviour using only two possible types of Izhikevich neuron. Changing the type of Izhikevich neuron may lead to different forecast functions. A general rule that used the neuron parameters a, b, c and d to evaluate the statistical function

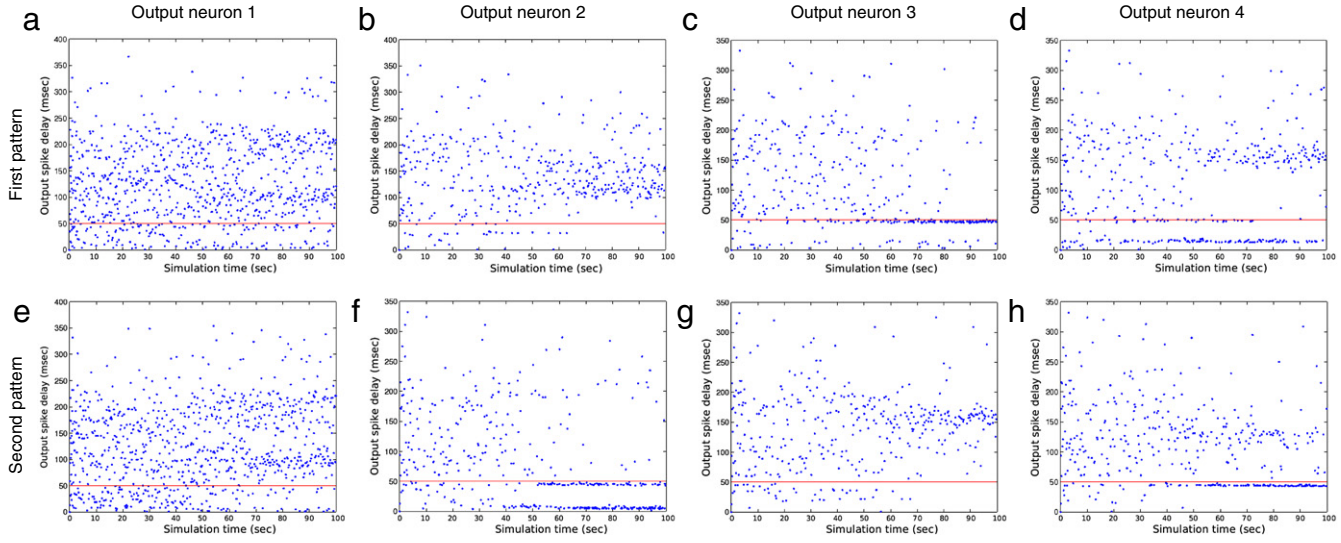


Fig. 16. Scatter plots for STDP with TTS forecast: two input patterns, four output neurons, simulated on the SpiNNaker system. The learning parameter is set to $L = -65$ mV. The first row is related to the first input pattern, while the second row is related to the second input pattern. Each of the columns refers to a single output neuron.

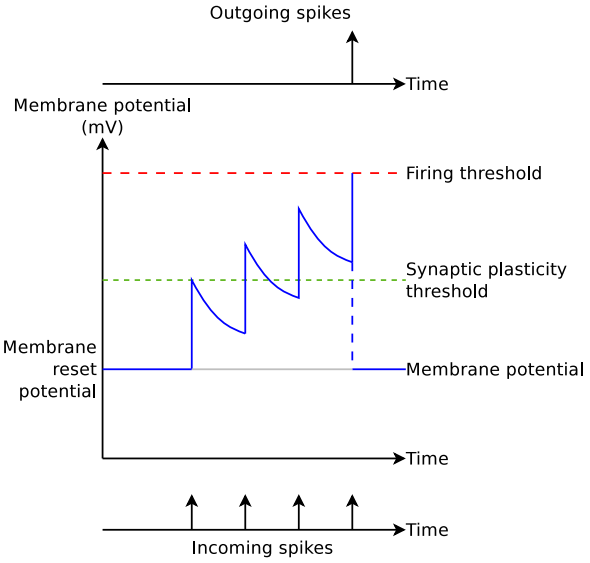


Fig. 18. Description of synaptic plasticity trigger. Only the last incoming spike triggers LTP because the membrane potential is above the plasticity threshold. Even if this synapse is strengthened the previous three synapses from which the three spikes are received are not influenced and the output spike cannot tune to the earlier spikes.

would be much more useful, but on the other hand there may be neurons for which no forecast function exists. Chaotic neurons may belong to this category, although for such neurons it is questionable whether they are able to learn to respond to specific input patterns. Systematic analysis of the relationship between L, a, b, c, and d will be important future work.

Thus far, the simulations have run on the current-generation SpiNNaker board, containing 4 SpiNNaker chips, and hence a total of 72 cores. While all simulations run in real-time, with the small size of network in these experiments, differences in execution speed are insignificant compared to simulation on a conventional PC platform. However, using the same board, larger simulations have been run that exhibit a dramatic performance increase over standard platforms. A prototype version of the board using 4 2-core chips for a combined 8 processors already showed significant increases (see Rast et al., 2011). Larger boards are currently in development having 48 chips, for a total of 864 cores, and dedicated board-to-board links to permit assembly into very large systems. With these boards we expect to begin testing of models with hundreds of thousands to millions of neurons, and to examine scalability systematically.

8. Conclusions

In this paper we have introduced and tested a new learning rule for artificial spiking neural networks based on a statistical approach. The learning efficacy of the algorithm has been tested against the standard STDP algorithm, showing comparable results in a pattern recognition task. A first implementation of the STDP TTS learning rule on the SpiNNaker neuromimetic architecture shows its advantages. Since the algorithm is triggered only when a spike is received, it is particularly suitable for event-driven systems where memory and computation are at a premium.

Acknowledgments

The SpiNNaker project is supported by the Engineering and Physical Science Research Council (EPSRC), grant EP/4015740/1, and also by ARM and Silistix. We appreciate the support of these sponsors and industrial partners.

References

Abbott, L. F., & Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nature Neuroscience*, *3*(Suppl.), 1178–1183.
Arena, P., Fortuna, L., Frasca, M., Patané, L., & Sala, C. (2007). Integrating high-level sensor features via STDP for bio-inspired navigation. In *Circuits and systems. 2007. ISCAS 2007. IEEE international symposium on. IEEE* (pp. 609–612).
Arthur, J.V., & Boahen, K. (2006). Learning in silicon: timing is everything. In B. Sholkopf, & Y. Weiss (Eds.), *Advances in Neural Information Processing Systems: Vol. 17* (pp. 281–1185). Vol. 18.
Artola, A., Brocher, S., & Singer, W. (1990). Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex. *Nature*, *347*(6288), 69–72.
Bi, G.-Q., & Poo, M.-M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of Neuroscience*, *18*(24), 10464–10472.
Bi, G., & Poo, M. (2001). Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual Review of Neuroscience*, *24*(1), 139–166.
Bienenstock, E. L., Cooper, L. N., & Munro, P. W. (1982). Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *The Journal of Neuroscience*, *2*(1), 32–48.
Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J., et al. (2007). Simulation of networks of spiking neurons: a review of tools and strategies. *Journal of Computational Neuroscience*, *23*(3), 349–398.
Burkitt, A. N., Gilson, M., & van Hemmen, J. L. (2007). Spike-timing-dependent plasticity for neurons with recurrent connections. *Biological Cybernetics*, *96*(5), 533–546.
Cajal, S. R. (1894). The Croonian lecture: La fine structure des centres nerveux. *Proceedings of the Royal Society of London*, *55*(331–335), 444–468.
Carpenter, G. A., & Grossberg, S. (1987). ART 2: self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, *26*(23), 4919–4930.
Clopath, C., Busing, L., Vasilaki, E., & Gerstner, W. (2010). Connectivity reflects coding: a model of voltage-based STDP with homeostasis. *Nature Neuroscience*, *13*(3), 344–352.
Davies, S., Rast, A., Galluppi, F., & Furber, S. (2011). A forecast-based biologically-plausible STDP learning rule. In *IJCNN 2011* (pp. 1810–1817).
Delorme, A. (2001). Networks of integrate- and-fire neurons using rank order coding b: spike timing dependent plasticity and emergence of orientation selectivity. *Neurocomputing*, *38–40*(1–4), 539–545.
Furber, S. B., Temple, S., & Brown, A. D. (2006). High-performance computing for systems of spiking neurons. In *Proc. AISB'06 workshop on GC5: architecture of brain and mind*.
Gerstner, W., & Kistler, W. M. (2002). Mathematical formulations of Hebbian learning. *Biological Cybernetics*, *87*(5–6), 404–415.
Gilson, M., Burkitt, A. N., Grayden, D. B., Thomas, D. A., & van Hemmen, J. L. (2009). Emergence of network structure due to spike-timing-dependent plasticity in recurrent neuronal networks IV. *Biological Cybernetics*, *101*(5–6), 427–444.
Grossberg, S., & Versace, M. (2008). Spikes, synchrony, and attentive learning by laminar thalamocortical circuits. *Brain Research*, *1218*, 278–312.
Guyonneau, R., VanRullen, R., & Thorpe, S. J. (2005). Neurons tune to the earliest spikes through STDP. *Neural Computation*, *17*(4), 859–879.
Hebb, D.O. (1949). *The organization of behavior: a neuropsychological theory*.
Indiveri, G., Chicca, E., & Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Transactions on Neural Networks*, *17*(1), 211–221.
Indiveri, G., Stefanini, F., & Chicca, E. (2010). Spike-based learning with a generalized integrate and fire silicon neuron. In *Circuits and systems, 2010. ISCAS 2010. IEEE international symposium on* (pp. 1951–1954).
Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 1569–1572.
Izhikevich, E. M. (2006). Polychronization: computation with spikes. *Neural Computation*, *18*(2), 245–282.
Izhikevich, E. M., & Desai, N. S. (2003). Relating STDP to BCM. *Neural Computation*, *15*(7), 1511–1523.
Jin, X., Rast, A., Galluppi, F., Davies, S., & Furber, S. (2010). Implementing spike-timing-dependent plasticity on SpiNNaker neuromorphic hardware. In *Neural networks, 2010. IJCNN 2010. IEEE world congress on computational intelligence. IEEE international joint conference on* (pp. 1–8).
Jin, X., Rast, A., Galluppi, F., Khan, M., & Furber, S. (2009). Implementing learning on the SpiNNaker universal neural chip multiprocessor. In C. S. Leung, M. Lee, & J. H. Chan (Eds.), *Neural information processing: Vol. 5863* (pp. 425–432) Berlin, Heidelberg: Springer (Chapter 48).
Kube, K., Herzog, A., Michaelis, B., de Lima, A. D., & Voigt, T. (2008). Spike-timing-dependent plasticity in small-world networks. *Neurocomputing*, *71*(7), 1694–1704.
Livi, P., & Indiveri, G. (2009). A current-mode conductance-based silicon neuron for address-event neuromorphic systems. In *2009 IEEE international symposium on circuits and systems* (pp. 2898–2901). IEEE.
MacNeil, D., & Eliasmith, C. (2011). Fine-tuning and the stability of recurrent neural networks. *PLoS One*, *6*(9), e22885.
Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Biological Cybernetics*, *14*(2), 85–100.
Markram, H., & Tsodyks, M. (1996). Redistribution of synaptic efficacy between neocortical pyramidal neurons. *Nature*, *382*(6594), 807–810.
Masquelier, T., Guyonneau, R., & Thorpe, S. J. (2008a). Competitive STDP-based spike pattern learning. *Neural Computation*, *21*(5), 1259–1276.

Masquelier, T., Guyonneau, R., & Thorpe, S. J. (2008b). Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLoS One*, *3*(1), e1377.

Masquelier, T., & Thorpe, S. J. (2007). Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Computational Biology*, *3*(2), e31.

Masuda, N., & Kori, H. (2007). Formation of feedforward networks and frequency synchrony by spike-timing-dependent plasticity. *Journal of Computational Neuroscience*, *22*(3), 327–345.

Migliore, M., Cannia, C., Lytton, W. W., Markram, H., & Hines, M. L. (2006). Parallel network simulations with NEURON. *Journal of Computational Neuroscience*, *21*(2), 119–129.

Pfister, J.-P. P., & Gerstner, W. (2006). Triplets of spikes in a model of spike timing-dependent plasticity. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, *26*(38), 9673–9682.

Plana, L. A., Furber, S. B., Temple, S., Khan, M., Shi, Y., Wu, J., et al. (2007). A GALS infrastructure for a massively parallel multiprocessor. *IEEE Design & Test of Computers*, *24*(5), 454–463.

Rao, R. P. N., & Sejnowski, T. J. (2001). Spike-timing-dependent Hebbian plasticity as temporal difference learning. *Neural Computation*, *13*(10), 2221–2237.

Rast, A., Galluppi, F., Davies, S., Plana, L., Patterson, C., Sharp, T., et al. (2011). Concurrent heterogeneous neural model simulation on real-time neuromimetic hardware. *Neural Networks*, *24*, 961–978.

Rast, A. D., Jin, X., Galluppi, F., Plana, L. A., Patterson, C., & Furber, S. (2010). Scalable event-driven native parallel processing: the SpiNNaker neuromimetic system. In *Proceedings of the 7th ACM international conference on computing frontiers*, CF'10. (pp. 21–30). New York, NY, USA: ACM.

Rast, A., Jin, X., Khan, M., & Furber, S. (2009). The deferred-event model for hardware-oriented spiking neural networks. In *Proc. 2008 int'l conf. neural information processing*, ICONIP 2008. (pp. 1057–1064). Springer-Verlag.

Rast, A., Navaridas, J., Jin, X., Galluppi, F., Plana, L., Miguel-Alonso, J., et al. (2011). Managing burstiness and scalability in event-driven models on the SpiNNaker neuromimetic system. *International Journal of Parallel Programming*, 1–30.

Roberts, P. D., & Bell, C. C. (2002). Spike timing dependent synaptic plasticity in biological systems. *Biological Cybernetics*, *87*(5–6), 392–403.

Schemmel, J., Brüderle, D., Meier, K., & Ostendorf, B. (2007). Modeling synaptic plasticity within networks of highly accelerated I&F neurons. In *Circuits and systems, 2007. ISCAS 2007. IEEE international symposium on. IEEE* (pp. 3367–3370).

Sejnowski, T. J. (1977). Statistical constraints on synaptic plasticity. *Journal of Theoretical Biology*, *69*(2), 385–389.

Sjöström, J., & Gerstner, W. (2010). Spike-timing dependent plasticity. *Scholarpedia*, *5*(2), 1362.

Song, S., & Abbott, L. F. (2001). Cortical development and remapping through spike timing-dependent plasticity. *Neuron*, *32*(2), 339–350.

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, *3*(9), 919–926.

Thorpe, S., & Gautrais, J. (1998). Rank order coding. In *Proceedings of the sixth annual conference on computational neuroscience: trends in research, 1998: trends in research, 1998*, CNS'97. (pp. 113–118). New York, NY, USA: Plenum Press.

Turrigiano, G. G. (1999). Homeostatic plasticity in neuronal networks: the more things change, the more they stay the same. *Trends in Neurosciences*, *22*(5), 221–227.

Wang, H.-X. X., Gerkin, R. C., Nauen, D. W., & Bi, G.-Q. Q. (2005). Coactivation and timing-dependent integration of synaptic potentiation and depression. *Nature Neuroscience*, *8*(2), 187–193.

Willshaw, D., & Dayan, P. (1990). Optimal plasticity from matrix memories: what goes up must come down. *Neural Computation*, *2*, 85–93.