

# An analysis of the use of Hebbian and Anti-Hebbian Spike Time Dependent Plasticity learning functions within the context of recurrent spiking neural networks.

Andrew Carnell  
 Dept of Computer Science  
 University of Bath  
 Bath, BA2 7AY, UK  
 Email: csparc@bath.ac.uk

**Abstract**—It is shown that the application of a form of Spike Time Dependent Plasticity (STDP) within a highly recurrent spiking neural net based upon the LSM, leads to an approximate convergence of the synaptic weights. Convergence is a desired property as it signifies a degree of stability within the network. An *activity link*  $L$  is defined which describes the link between the spiking activity on a connection and the weight change of the associated synapse. It is shown that under specific conditions Hebbian and Anti-Hebbian learning can be considered approximately equivalent. Also, It is shown that such a network habituates to a given stimulus and is capable of detecting subtle variations in the *structure* of the stimuli itself.

## I. THE LIQUID STATE MACHINE

The Liquid State Machine (LSM) concept was introduced by Maass *et al* and for full details see [6]. The LSM in one of its basic form consists of a pool of highly recurrently connected spiking *Leaky Integrate and Fire* (LIF) neurons.

This pool typically receives a spiking input  $n_{input}(t)$  from a pool of input neurons that are connected to a selection of neurons within the LSM. The LSM acts as a medium through which the input can be expressed in a higher dimensional form. A readout neuron which receives a connection from all of the neurons within the LSM is then able to be taught, using a learning algorithm on the connections from the pool, to perform some computable function  $F(n_{input}(t))$  on the input, with the strong limiting factor being the size of the LSM, see [1].

### A. A Typical Implementation

Analyses of the computational qualities of the LSM typically involve the use of a randomly generated layer of LIF neurons which facilitate the projection of an input into higher dimensions, as shown in Fig. 1. An input spike train  $n_{input}(t)$  is presented to the ‘liquid’ layer. Learning of a particular function  $F(n_{input}(t))$  is then typically accomplished by altering the synaptic weights that connect a readout pool of neurons to this liquid layer. For example, using some learning algorithm it is, in some cases, possible to train a pool of readout neurons to extract from the liquid layer, the information about the

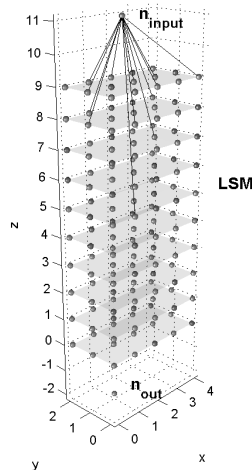


Fig. 1. A Basic LSM consisting of LIF neurons with spiking input neuron  $n_{input}$  and LIF readout neuron  $n_{out}$

input stream  $n_{input}(t)$  up-to some time into the past  $\tau$ , that is required to compute a given function  $F(n_{input}(t - \tau))$  for different values of  $\tau$ . The difference in implementation in this paper is that unlike the LSM, which typically has static weights in the recurrent part of the network, the effect of using dynamic weights is investigated.

## II. SPIKE TIME DEPENDENT PLASTICITY LEARNING

Spike Time Dependent Plasticity or STDP learning rules are based upon Hebb’s postulate, see [3] which can written informally as: neurons that fire together, wire together. Suppose that two neurons are connected, and that if the pre-synaptic neuron  $n_{pre}$  is active (spiking) before the post-synaptic neuron  $n_{post}$ , then the synaptic weight on the link between them is strengthened. In this case  $n_{pre}$  can be thought of as perhaps contributing to the firing of  $n_{post}$ , so its influence is encouraged. This is known as Hebbian STDP. If the firing sequence is reversed and  $n_{pre}$  fires *after*  $n_{post}$

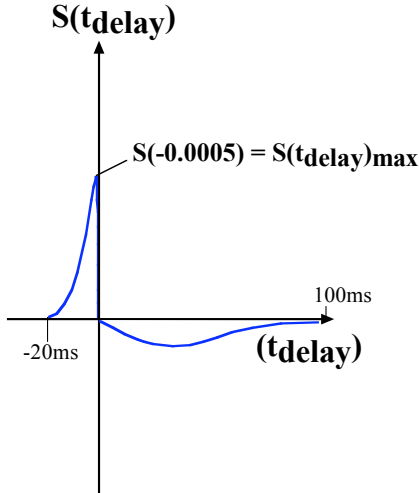
then the connection is weakened.

The case when the synaptic weight is strengthened if  $n_{post}$  fires before  $n_{pre}$  and weakened if the firing is reversed is known as Anti-Hebbian STDP. The delay between the firing time  $t_{pre}$  of  $n_{pre}$  and the firing time  $t_{post}$  of  $n_{post}$  is denoted as  $t_{delay} = t_{pre} - t_{post}$  and is the determining factor in how large the synaptic weight change should be.

### A. The STDP Learning Window Functions

In this paper, an asymmetric Hebbian learning window function is used, the form of which can be seen in Fig. 2. This is a typical learning window function which increases those synaptic weights whose pre-synaptic neuron fires just before its post-synaptic neuron and weakens the synaptic weight if the post-synaptic neuron fires before the pre-synaptic neuron. The amount of the modification is determined according to the product of the learning window function  $S(t_{delay})$  and some learning rate  $\phi$ .

## Hebbian learning function



$$\begin{aligned} S(t_{delay}) &= 0.08 * e^{(t_{delay}/0.005)} && \text{for } 20ms \leq t_{delay} \leq 0.5ms \\ S(t_{delay}) &= -t_{delay} * e^{(t_{delay}/0.0002)^2} && \text{for } 0.5ms < t_{delay} \leq 0ms \\ S(t_{delay}) &= -0.02 * e^{(-t_{delay}/0.03)} && \text{for } 0ms < t_{delay} \leq 100ms \end{aligned}$$

Fig. 2. The form of the learning window function used to calculate the weight change due to STDP is shown in Fig. 2 above. The actual weight update value is given by the product of this learning function  $S(t_{delay})$  and the learning rate  $\phi$ . Typically  $\phi = 2 * 10^{-7}$ , and was determined experimentally.

An Anti-Hebbian learning window function is one that operates in reverse to the Hebbian version. Therefore, synaptic weights are strengthened if the post-synaptic neuron fires before the pre-synaptic neuron, and weakened if the pre-synaptic neuron fires before the post-synaptic. In this implementation, for Anti-Hebbian STDP  $-\phi$  is used for the learning rate instead of  $\phi$ .

**Note :** The time scale of the learning window function is such, that the peak occurs at a distance of  $0.5ms$  from the post-synaptic firing time. This value was chosen in accordance with the finding by Gerstner in [2], that the learning window

function maximum  $S(t_{delay})_{max}$  should be in the range  $\delta/2 \leq S(t_{delay})_{max} \leq \delta$ , where  $\delta$  is the rise time of a post-synaptic spike, typically  $0.5ms$ . STDP is a biologically derived learning method, but there are many unanswered questions as to what actually happens when STDP is applied, especially in the context of recurrent spiking neural networks.

### B. Weight Change and Normalisation

Consider a neuron  $Y$  within a recurrent neural network, similar to Fig 1, that receives inputs from neurons  $X_1, \dots, X_k$  with respective synaptic weights  $w_1 \dots w_k$ .  $W$  is the weight vector associated with the input of  $Y$ ,  $W = (w_1 \dots w_k)$ . The individual synaptic weights that comprise  $W$  are modified by two processes; a Hebbian weight update that is calculated for  $w_1 \dots w_k$ , and a normalisation procedure which ensures that the input synapses to a neuron have a constant norm, therefore this a competitive Hebbian process. The normalisation is applied to  $W$  only after all weights  $w_1 \dots w_k$  have had the Hebbian weight update procedure applied. Network activity is simulated in  $0.5s$  time-steps for the duration of the simulation. These update procedures are performed on the weight vectors of each neuron of the network at each time-step. Suppose  $R$  is the norm of the original unmodified weight vector  $W$ . For a neuron  $X_i$  firing at time  $t_i$  and a neuron  $Y$  firing at time  $\tau$ , we get:

$$\begin{aligned} w_i &:= w_i + \phi S(t_i - \tau) \text{ for } i = 1, \dots, k \\ W &:= R * W / \|W\| \end{aligned}$$

### C. Activity Link

In order to better describe the weight change brought about by the Hebbian update, and as a means for describing the link between the spiking activity on a synaptic connection and the weight change of the synapse associated with that connection, the activity link  $L(a, b)$  is defined where  $a$  and  $b$  are spike trains. Suppose that  $a$  has spike times  $t_1, \dots, t_m$  and  $b$  has spike times  $\tau_1, \dots, \tau_n$ , then we can define the following:

$$L(a, b) = \sum_{i=1}^m \sum_{j=1}^n S(t_i - \tau_j)$$

Consider neuron  $Y$  from the previous section. Suppose that the spike trains of  $X_1, \dots, X_K$  are  $x_1, \dots, x_k$  and  $Y$  has a spike train  $y$  over an interval  $I$ . The update rules for the Hebbian learning followed by the normalisation update are:

$$\begin{aligned} w_i &:= w_i + \phi L(x_i, y) \text{ for } i = 1, \dots, k \\ W &:= R * W / \|W\| \end{aligned}$$

## III. LSM GENERATION PARAMETERS

All experiments are performed using CSIM under MATLAB. CSIM allows for the simulation of many types of neuron and synapse models, and enables the creation of pools of recurrently connected neurons. It was developed by the Neural Micro-Circuits group at *Technische Universitat Graz*, and the software and manuals can be found at <http://www.lsm.tugraz.at/csim/index.html>.

For the sake of completeness, the choice of the major parameters are given below.

**ConnectionProbability**– This is determined by the term  $C \cdot e^{D(a,b)/\lambda}$ :  $\lambda$  determines the average distance between neurons as well as the average number of connections, the euclidean distance between two neurons  $a$  and  $b$  is given by  $D(a, b)$ , and the value of  $C$  is dependent on whether  $a$  and  $b$  are excitatory (E), or inhibitory (I). Therefore  $C$  is 0.3(EE), 0.2 (EI), 0.4 (IE), 0.1(II). This is taken from Maass *et al*, see [5].

**PostSynapticSpike**– This is of the form  $e^{(-t/\tau)}$ .  $\tau = 3ms$  for excitatory and  $\tau = 6ms$  for inhibitory synapses.

#### IV. EXPERIMENTS

##### A. Experimental Setup

A recurrently connected network of 150 LIF neurons in a  $5 \times 3 \times 10$  arrangement is created with connectivity parameters  $C = 3$  and  $\lambda = 2$ . A single spiking input neuron is connected to a random selection the 150 total neurons that comprise the network. The connectivity parameters for connecting the input to the network are given the values  $C = 3$  and  $\lambda = 3$  and the input neuron is positioned so as to cause it to connect to between 10% and 20% of the recurrent network neurons. An input spike train is generated as follows. For each time-bin there is a probability of 0.5 of getting a burst of spikes at a frequency of 150Hz, and a probability of 0.5 of getting no spikes at all. The time-bin duration is 20ms. All experiments use this setup for the network and input generation unless stated otherwise.

##### B. Synaptic Weight Convergence

10 Networks were generated using the above parameter values. Hebbian STDP was then applied to the internal weights of the network while the network was stimulated by the input neuron for a period of 300s. For each of the 10 networks and subsequent to the Hebbian learning just described, a form of Anti-Hebbian learning was applied to the network. also for a period of 300s. Again the average weight change over the whole network was recorded. One might expect that this would have the effect of reversing the work done by the Hebbian learning. However it appears that this is not the case. Instead what is observed is that once network weight changes due to Hebbian learning have reached a stable state, the subsequent application of Anti-Hebbian learning would appear to continue to allow the network to exist in this same stable state.

This observation can be explained by using the concept of the activity link defined in section 2 and can be shown to be the case by further experimentation. The input weight vector of each neuron in the recurrent network is bounded by the normalisation procedure described in section 2. As an analogy, the weight vector for each neuron in the recurrent network, can be said to be constrained to a multidimensional sphere of radius  $\|W\|$ , see Fig. 3. The ‘orientation’ of a weight vector is affected by changes to its constituent individual synaptic weights.

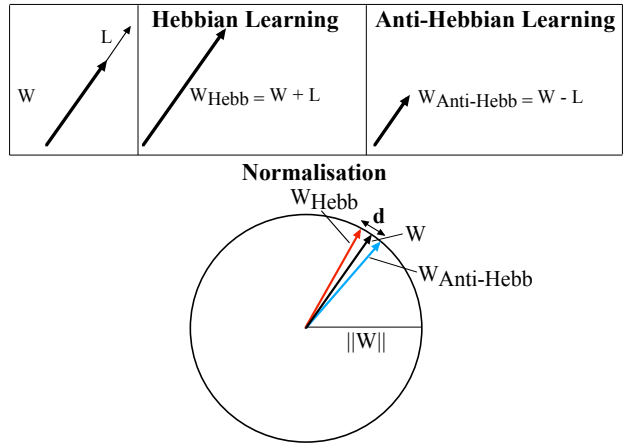


Fig. 3. An illustration of the how Hebbian and Anti-Hebbian learning are approximately equivalent, i.e.  $d$  is small, if  $W$  and  $L$  are approximately ‘aligned’ and if the learning rate  $\phi$  is small.

Recall neuron  $Y$  and its input spike trains  $x_1, \dots, x_k$ . Figure 3 illustrates the weight vector of  $Y$ ,  $W = (w_1, w_2, \dots, w_k)$  when it is modified under Hebbian and Anti-Hebbian learning. In the case of Hebbian learning,  $W$  is updated as described in section 2 (before normalisation). A small value of learning rate  $\phi$  is assumed. It is important the  $\phi$  is not too large as this would cause synaptic weight changes to become erratic, and meaningful changes would be less likely to occur.

An approximate fixed point/stable state, of this Hebbian process is defined as being when the weight vectors of all neurons are approximately aligned with their activities. For neuron  $Y$  this means  $W$  is approximately aligned with the activity link vector  $L = (L(x_1, y), L(x_2, y), \dots, L(x_k, y))$ . Given this definition of fixed points and of the activity link it can be seen that when  $\phi$  is small and  $W$  and  $L$  are approximately in the same ‘direction’, the modifications to  $W$  caused by Hebbian learning followed by normalisation are approximately equivalent to modifications by Anti-Hebbian learning.

The variable  $d$  in Fig. 3 describes the ‘difference’ between  $W$  after Hebbian and  $W$  after Anti-Hebbian learning.  $d$  will tend to be small when  $W$  and  $L$  are approximately in the same direction and if  $\phi$  is sufficiently small, in which case Anti-Hebbian learning will have the same approximate stable states as Hebbian learning for a given recurrent network. It is supposed that for any single network there may exist many of these stable states, each of which could be maintained by Hebbian or Anti-Hebbian learning.

To demonstrate experimentally, the necessity for the synaptic weights to be aligned with the activities on their connections in order for Hebbian and Anti-Hebbian learning to be able to be considered approximately equivalent, three networks were generated as before, stimulated with an input spike train and Hebbian learning was applied for 100 time-steps for the first network, 1000 time-steps for the second network and 5000 time-steps for the third. For each network, subsequent to the application of Hebbian STDP, Anti-Hebbian

STDP was then applied for the same number of time-steps. The results of this can be seen in Fig. 6.

It can clearly be seen that in the case of 100 time-steps exposure to Hebbian STDP, the subsequent application of Anti-Hebbian learning provokes a large change in the mean synaptic weights of the network, similar to the initial change provoked by the Hebbian learning. It is supposed that this is due to the synaptic weights not yet being sufficiently aligned with the activities on their connections because of insufficient training time - only 100 time-steps.

In the case where the network was trained for 1000 time-steps, a large change in mean weights is also observed when Anti-Hebbian STDP is applied after the Hebbian STDP. However, it can be seen that the magnitude of these initial changes is not as great as those seen in the 100 time-step case.

In the final example, in which each phase of STDP training is 5000 time-steps in duration, it can be seen that the application of Anti-Hebbian STDP provokes no weight change that is significantly different to the weights changes that the network was experiencing at the end of the Hebbian STDP training. At this point it is supposed that the network has undergone exposure to Hebbian STDP for a sufficient enough duration such that the majority of synaptic weights within the recurrent network are now aligned with their activities and so the changes brought about to the network by Hebbian and Anti-Hebbian STDP are now approximately equivalent.

### C. Hebbian Learning to Demonstrate Habituation

Using the same experimental setup as in the previous experiment, the stimulus spike train is now generated such that its frequency varies as a function of time. More specifically, the number of spikes,  $f$ , that occur in any 20ms time bin of the input spike train, is given by  $f = C \cdot \sin(t/t_c) + C$ . Where  $t_c$  is the time constant of the function, and  $C$  is a scalar which determines the range over which  $f$  varies. A spike train,  $S_1$ , is created according to  $f_1 = 4 \cdot \sin(t/2) + 4$ . This stimulus is then used to drive the recurrent network to which the Hebbian learning window function is applied. The mean change in synaptic weights over the whole network is calculated and recorded at each 0.5s time step as before. The network is subjected to this stimulus for a duration of 200 seconds.

A second input spike train,  $S_2$ , is now created using a similar function to the first but with a time constant  $t_c = 10$ . So,  $f_2 = 4 \cdot \sin(t/10) + 4$ . The same recurrent network that was subjected to  $S_1$  for 200 seconds is now subjected to  $S_2$  also for 200 seconds and again the mean change in synaptic weight over the whole network is recorded at each time step. It can be seen in the top left panel of Fig. 7 that the mean weight change undergoes a large initial change and then settles down. At the onset of the change in the frequency modulation of the stimulus from  $S_1$  to  $S_2$  (time-step 400) it can be seen that the mean weight change once again undergoes a sharp increase and again settles down. If this same network is then re-presented with  $S_1$ , see top right panel of Fig. 7, the network

does not respond with any kind of sharp change in synaptic weights. Similarly if the network is again presented with a stimulus drawn from  $S_2$  (from time-step 400 in top right panel of Fig. 7), the sharp change in weights is also absent. Figure 4 shows a plot of  $f_1$  and  $f_2$ .

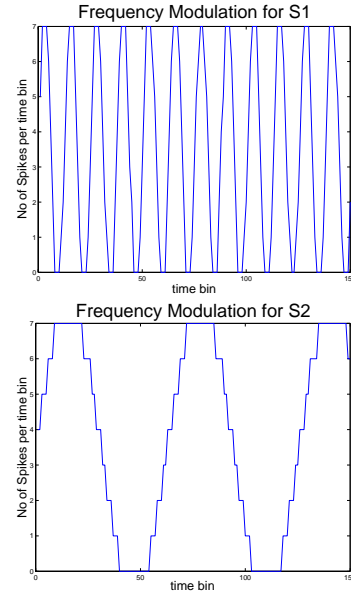


Fig. 4. Plots of the different frequency modulation functions  $f_1$  and  $f_2$  used to create the input spike trains  $S_1$  and  $S_2$ . The left panel shows a plot of  $f_1 = 4 \cdot \sin(t/2) + 4$ . The right panel shows a plot of  $f_2 = 4 \cdot \sin(t/10) + 4$ .

This is an interesting occurrence because it would appear that upon being subjected to the stimulus with frequency modulation  $f_1$  the network becomes habituated to this stimulus after prolonged exposure. When the frequency modulation is changed to  $f_2$ , the network is able to detect this subtle change and responds with an alteration of synaptic weights so that the weight vectors become aligned with the activity links once again. However, the weights are apparently altered in such a way that the learning of the frequency modulation of  $S_1$  is not forgotten. This can be seen in the top right panel of Fig. 7 where  $S_1$  doesn't provoke another series of sharp changes to the mean weight. This effect was observed by frequency modulation on the spike train stimulus of only a single spiking input neuron. Future work will extend this to investigate how many frequency modulations a network of given size can habituate to from a single input neuron. Additionally, it is thought that the introduction of multiple input neurons will increase number of 'Habituations' a given network could learn.

It can be seen in the lower panel in Fig. 7 that, as the difference between the time constants of the two input decreases, i.e. the two stimuli become more similar, the network is less able to respond in a distinct manner to each stimulus as expected. However, it is not yet known precisely how subtle a change in input stimuli is detectable by a given network. This is a topic of further investigation.

## V. RESULTS

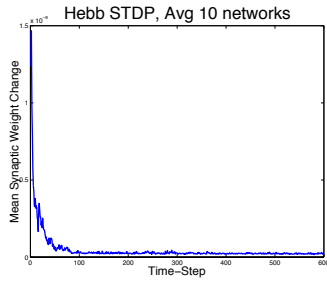


Fig. 5. A graph of the absolute mean synaptic weight change due to Hebbian STDP, averaged over 10 different. The network is exposed to a stimulus of duration 300s.

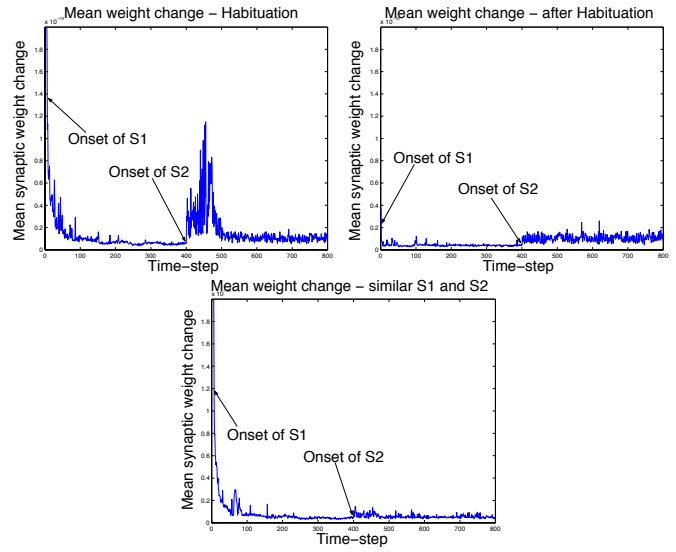


Fig. 7. The top left panel of Fig. 7 shows the variation of the absolute mean weight change when the network is presented with S1 and then S2. The top right panel shows the mean weight change for the same network for S1 and then S2 after exposure to S1 and S2 in the top left panel. The lower panel is a plot of the mean weight change when the time constants of the frequency modulation of S1 and S2 are similar in value ( $t_c = 2$  for S1 and  $t_c = 2.5$  for S2).

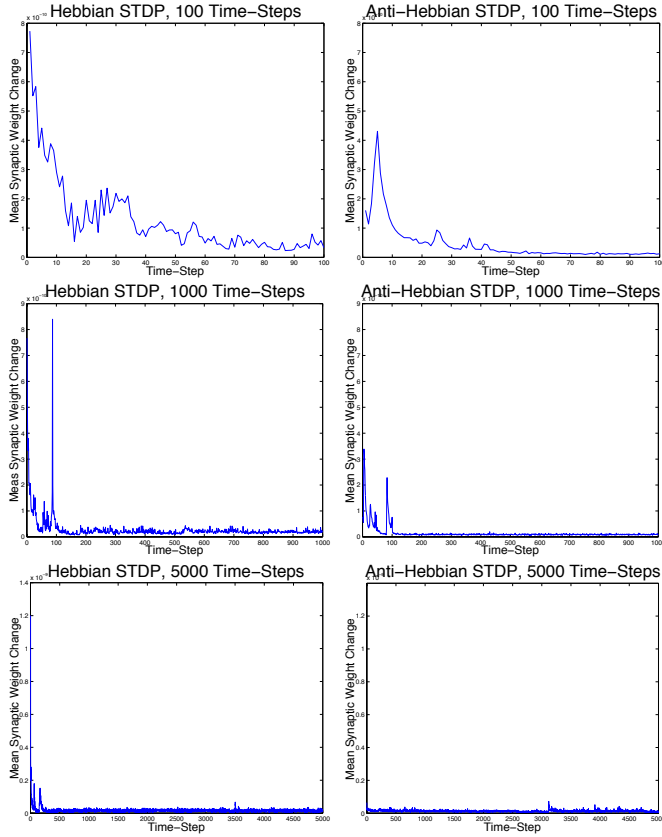


Fig. 6. The top pair of panels show Hebbian and then Anti-Hebbian STDP on a network each for a duration of 100 time-steps. The middle and lower pair show the same plots for different networks and for exposures of 1000 and 5000 time-steps respectively.

## VI. DISCUSSION

The results shown in Fig. 5, clearly demonstrate that the version of STDP Hebbian learning outlined in section 2, when applied with appropriate learning parameters, facilitates the approximate convergence of the synaptic weights of the recurrent network. In such a case, the weight vector of each neuron becomes aligned with the activity on its own inputs.

This means that the network has reached some stabilised state where the values of the synaptic weights have been determined by the spiking input activity into the network. Figure 5 shows mean weight change data averaged over 10 networks. In each case the convergence is seen, even though the networks are generated in a random manner. This would seem to indicate that the convergence process is quite robust to variations in network connectivity.

An interesting result is that of the equivalence of Hebbian and Anti-Hebbian learning, as shown in Fig. 6. In section 4 it was said that in order for Hebbian and Anti-Hebbian STDP to be considered equivalent, two conditions should be met. Firstly, a small learning rate  $\phi$  should be used, and secondly, all synaptic weights should be approximately aligned with the spiking activity they receive. Using a learning rate of  $\phi = 2 \times 10^{-7}$ , Fig. 6 shows the result of exposing the network to a stimulus whilst performing STDP for progressively increasing durations. It can be seen in the top pair of panels of Fig. 6, that for 100 time steps the initial weight changes provoked by the Anti-Hebbian STDP are not similar to the changes seen at the end of the Hebbian STDP experiment. Similarly for the 1000 time step case. However, for 5000 time-steps of exposure, the Anti-Hebbian learning appears to continue on from the Hebbian learning. It is supposed that at the end of the Hebbian STDP phase, the network has reached a fixed point. Once one of these fixed points/stable states has been reached, as explained in section 4, the changes to a weight vector due to Hebbian or to Anti-Hebbian STDP can be considered approximately equivalent. This is an interesting result as it is perhaps counter to what one might expect to happen.

The Habituation experiments appear to show that it is possible, by applying the STDP learning functions described in this paper to a recurrent network of spiking LIF neurons, for a network to become habituated to a very specific spike train *structure* from a single spiking input neuron. Figure 7 shows that such a network then responds to subtle changes in the structure of the spike train on the *same* spiking input neuron by ‘aligning’ the synaptic weights with the new spiking activities. Additionally, It would appear that this happens without ‘erasing’ the weight changes which allowed for the Habituation of the initial spike train. There is much more research to be done on this subject and future research will involve investigating the capabilities of such a learning approach in terms of the sensitivity to structural changes in the input stimuli itself and the number of stimuli that can be learned before the network begins to ‘forget’ previously learned stimuli.

#### REFERENCES

- [1] N. Bertschinger, T. Natschlager, Real-Time Computation at the Edge of Chaos in Recurrent Neural Networks, *Neural Comp*, 16: 1413-1436, 2004.
- [2] W. Gerstner, W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, Cambridge University Press, Cambridge, 2002.
- [3] D.O. Hebb, *The organization of behavior: A neuropsychological theory*, Wiley, New York, 1949.  
Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. New York: Wiley.
- [4] G.E. Hinton, T.J. Sejnowski, Learning and Relearning in Boltzmann Machines, in J. L. McClelland, D. E. Rumelhart, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, MIT Press, Cambridge, MA, 1986.
- [5] W. Maass, T. Natschlager, H. Markram, Computational Models for Generic Cortical Microcircuits, *Computational Neuroscience: A Comprehensive Approach*, chapter 18, pages 575-605. Chapman & Hall/CRC, Boca Raton, 2004.
- [6] T. Natschlager, W. Maass and H. Markram, The “liquid computer”, a novel strategy for real-time computing on time series, *Special issue on Foundations of Information Processing of TELEMATIK*, 8 (1): 32-36, 2002.
- [7] T. Natschlager, W. Maass and H. Markram, Real Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations, *Neural Computation*, 14(11):2531-2560, 2002.
- [8] C.E. Shannon, A mathematical theory of communication (parts I and II). *Bell System Technical Journal*, XXVII:379-423, 1948.