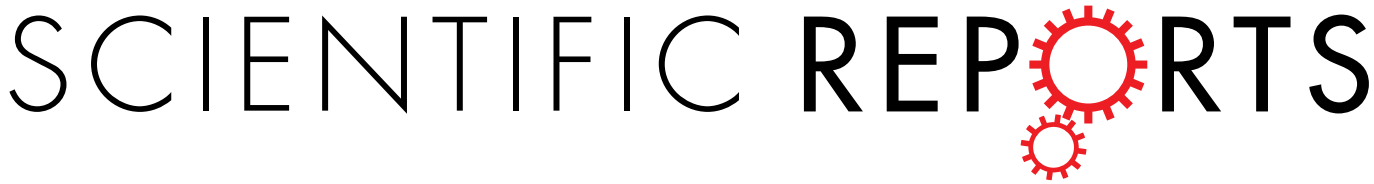


OPEN

Seven neurons memorizing sequences of alphabetical images via spike-timing dependent plasticity

Received: 20 May 2015
Accepted: 19 August 2015
Published: 16 September 2015

Takayuki Osogami* & Makoto Otsuka*

An artificial neural network, such as a Boltzmann machine, can be trained with the Hebb rule so that it stores static patterns and retrieves a particular pattern when an associated cue is presented to it. Such a network, however, cannot effectively deal with dynamic patterns in the manner of living creatures. Here, we design a dynamic Boltzmann machine (DyBM) and a learning rule that has some of the properties of spike-timing dependent plasticity (STDP), which has been postulated for biological neural networks. We train a DyBM consisting of only seven neurons in a way that it memorizes the sequence of the bitmap patterns in an alphabetical image "SCIENCE" and its reverse sequence and retrieves either sequence when a partial sequence is presented as a cue. The DyBM is to STDP as the Boltzmann machine is to the Hebb rule.

Artificial neural networks have been studied as means of automatic pattern recognition[1–3] for a long time, and the recent breakthrough of deep learning[4–8] has brought them once again to the forefront of artificial intelligence studies. A characteristic of an artificial neural network is associative memory, which stores multiple patterns in such a way that a particular pattern can be retrieved when it is given a cue such as a partial pattern[9–12]. The Hebb rule[13] is used to train artificial neural networks, including Perceptrons[14], Hopfield networks[10], and Boltzmann machines[15]. In particular, it incrementally decreases the energy of the patterns to be stored in a Boltzmann machine, which in turn increases the likelihood that the Boltzmann machine generates those patterns[16,17].

The Hebb rule used for artificial neural networks, however, has a fundamental shortcoming as a learning rule of biological neural networks, because the concept of time is largely missing from it. Specifically, it is independent of the precise timing of the spikes of neurons. A postulate that extends the Hebb rule is spike-timing dependent plasticity, or STDP[18–21], which states that a synapse is strengthened if the spike of a pre-synaptic neuron precedes the spike of a post-synaptic neuron (*i.e.*, long term potentiation; LTP[22,23]), and the synapse is weakened if the temporal order is reversed (*i.e.*, long term depression; LTD). The existence of STDP was experimentally confirmed around the end of the last century[24–26].

Here, we provide underpinnings for STDP as a learning mechanism by training an artificial neural network via STDP in such a way that the trained network exhibits associative memory for sequential patterns. Specifically, we model the dynamics of a biological neural network with an artificial neural network, which we refer to as a dynamic Boltzmann machine (DyBM). We train the DyBM by using an online learning rule that has some of the properties of STDP such as LTP and LTD. We sequentially present patterns to it and update its learnable parameters every time a pattern is presented. The DyBM then retrieves a particular sequence when associated cues are presented.

The structural features that distinguish a DyBM from a Boltzmann machine are conduction delays and memory units, which are illustrated in Fig. 1. A neuron is connected to another in a way that a spike from a pre-synaptic neuron, i, travels along an axon and reaches a post-synaptic neuron, j, via a synapse

IBM, IBM Research - Tokyo, Tokyo, 103-8510, Japan. *These authors contributed equally to this work. Correspondence and requests for materials should be addressed to T.O. (email: osogami@jp.ibm.com)

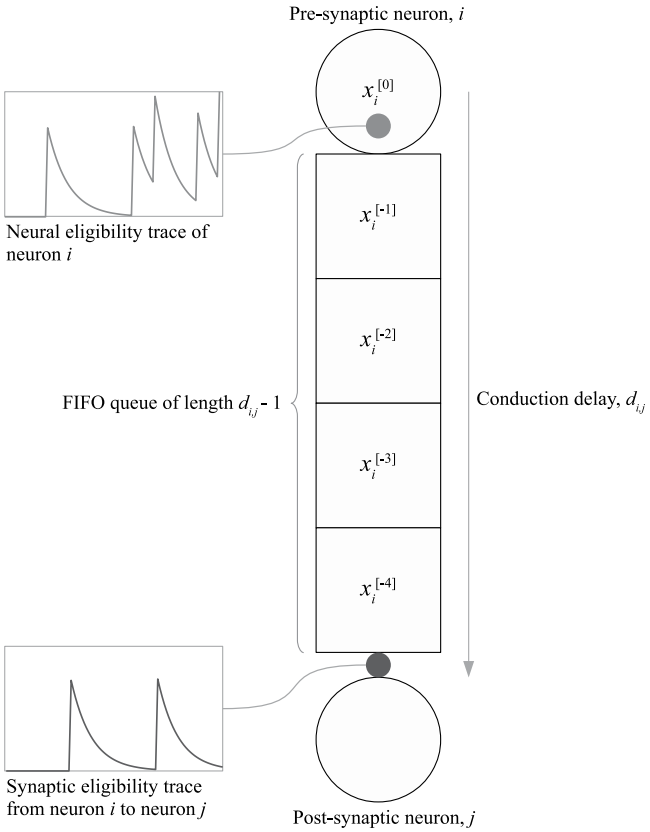


Figure 1. A DyBM consists of a network of neurons and memory units. A pre-synaptic neuron is connected to a post-synaptic neuron via a FIFO queue. The spike from the pre-synaptic neuron reaches the post-synaptic neuron after a constant conduction delay. Each neuron has the memory unit for storing neural eligibility traces, which summarize the neuron's activities in the past. A synaptic eligibility trace is associated with a synapse between a pre-synaptic neuron and a post-synaptic neuron, and summarizes the spikes that have arrived at the synapse, via the FIFO queue, from the pre-synaptic neuron.

after a delay consisting of a constant period, $d_{i,j}$. In the DyBM, a first-in first-out (FIFO) queue causes this conduction delay. The FIFO queue stores the values of the pre-synaptic neuron for the last $d_{i,j} - 1$ units of time. Each stored value is pushed one position toward the head of the queue when the time is incremented by one unit. The value of the pre-synaptic neuron is thus given to the post-synaptic neuron after the conduction delay. Moreover, the DyBM aggregates information about the spikes in the past into neural eligibility traces and synaptic eligibility traces[27–29], which are stored in the memory units. The value of a neural eligibility trace increases when an associated neuron spikes and decreases otherwise. The value of a synaptic eligibility trace increases when the spike from a pre-synaptic neuron reaches a post-synaptic neuron and decreases otherwise. In the current study, we keep three eligibility traces with varying decay rates for each neuron and for each synapse. The use of varying decay rates is consistent with the approximation of the hyperbolic decay for long-term memory[30–32].

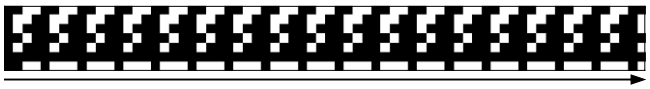
Each neuron takes a binary value, 0 or 1, and the probability that a neuron takes the value 1 (*i.e.*, it spikes) at any moment depends on the previous values of the neurons as well as the values of the learnable parameters of the DyBM. Each neuron is associated with a learnable parameter called bias. The strength of the synapse between a pre-synaptic neuron and a post-synaptic neuron is represented by learnable parameters called weights. In order to represent LTP and LTD, the DyBM has an LTP weight and an LTD weight. We use an online gradient ascent method[33,34] so as to maximize the likelihood of given sequential patterns. The learnable parameters are updated only on the basis of the information that is available at the associated synapse or neuron, and there is no need to store the whole sequence for learning via backpropagation through time[35–40].

Results

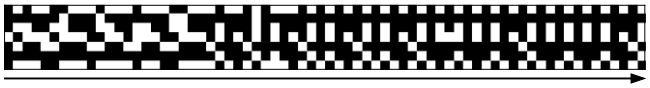
We trained a DyBM, consisting of seven neurons, in such a way that it could store multiple sequential patterns from alphabetical images. The DyBM would then retrieve a particular sequential pattern when part of it was presented as a cue, and it could also detect anomalies in a given sequence.

First, we trained the DyBM with a single sequence of patterns from an alphabetical image, in this case "SCIENCE," which is a 7-bit by 35-bit monochrome bitmap image. Each set of seven vertically aligned

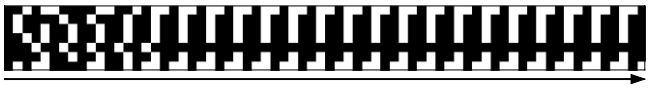
(a) Before training



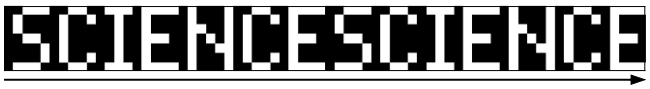
(b) After 10 periods of training



(c) After 1,000 periods of training



(d) After 100,000 periods of training



(e) After 130,000 periods of training

Figure 2. The DyBM learned the target sequence. (**a**) Before training began, the DyBM generated a sequence that was determined by the initial values of the learnable parameters. (**b–d**) In each period of training, we presented one period of the target sequence once to the DyBM. The DyBM gradually learned the target sequence as the training progresses from 10 periods to 100,000 periods. (**e**) After 130,000 periods, the DyBM generated the complete sequence.

bits composed the pattern of the moment, and the sequence of 35 of these patterns composed the period (see Fig. 2e). We presented each pattern one at a time in sequence to the DyBM and updated its learnable parameters each time. We also updated the values of the eligibility traces and the FIFO queues each time a 7-bit pattern was presented. Each training period consisted of presenting one period of the sequence (*i.e.*, presenting "SCIENCE" once). We repeated the training period multiple times.

To show the progress of training in Fig. 2, we let the DyBM generate a sequence, for two periods of the target sequence, before and after the training as well as after some intermediate steps. Before training began (see Fig. 2a), the DyBM generated a sequence that had nothing to do with the target sequence, because the initial values of the learnable parameters were independent of the target sequence. Here, we let the DyBM generate a sequence in a deterministic manner. At each moment, a neuron spiked if and only if the probability that the neuron spikes was greater than 0.5. This corresponds to making the temperature (see Supplementary Table S1a) of the DyBM infinitesimally small. Although the values of the learnable parameters were fixed when the DyBM was generating a sequence, the eligibility traces and the FIFO queues were updated on the basis of the generated sequence. The DyBM thus generated varying patterns during the two periods.

After 10 to 100,000 periods of training (see Fig. 2b–d), the DyBM generated a sequence whose first five 7-bit patterns composed part of "S." Here, we let the DyBM start generating a sequence without refreshing the eligibility traces or the FIFO queues that were updated during the training. The sequence presented during the training thus served as a cue for generating the sequence shown in Fig. 2. Training the DyBM completed in 85 seconds after 130,000 periods (see Fig. 2e), at which point the DyBM generated the complete sequence of the target pattern.

Figure 3 illustrates how the learnable parameters of the DyBM were updated during the training. The seven vertically aligned circles in a box represent the seven neurons, which are arranged from the top to the bottom in the order corresponding to the 7-bit patterns. The color of the circles represents the value of the associated bias in accordance with the color map shown in Fig. 3f. The neurons corresponding to the red circles are more likely to spike than others if the other conditions are equivalent. The arrows in Fig. 3 represent the LTP weights. As the conduction delay from pre-synaptic neuron i to post-synaptic neuron j is denoted by $d_{i,j}$, we draw an arrow from the i-th circle in the column labeled $-d_{i,j}$ to the j-th

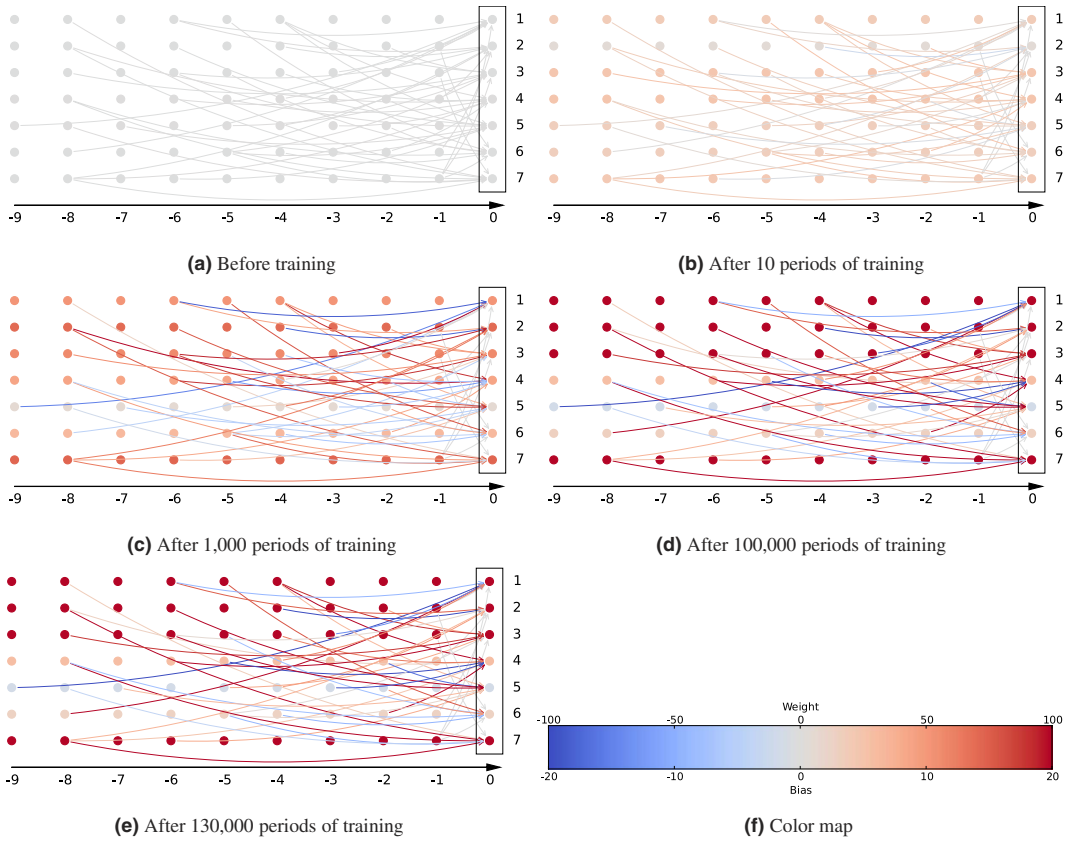


Figure 3. The DyBM updated the values of its learnable parameters during training with "SCIENCE." The color of a circle in a box shows the bias of a neuron. The color of an arrow shows the LTP weight ($u_{i,j,1} + u_{i,j,2} + u_{i,j,3}$ with the notations in Supplementary Table S1b) from a pre-synaptic neuron, i, to a post-synaptic neuron, j. The conduction delay from neuron i to neuron j is denoted by $d_{i,j}$, and an arrow is drawn from the i-th circle in the column labeled $-d_{i,j}$ to the j-th circle in the box. Here, the conduction delay is sampled independently from the uniform integer distribution with support [1, 9] (see Supplementary Table S2). (**a**) The initial values of the learnable parameters were sampled independently from the normal distribution with mean 0.0 and standard deviation 0.1. (**b**–**e**) Some of the learnable parameters were strengthened while others were weakened as the training progresses from 10 periods to 130,000 periods. The color map in (**f**) denotes the values of the learnable parameters.

circle in the box. The conduction delay was sampled independently from an integer uniform distribution between 1 and 9 (see Supplementary Table S2). The color of an arrow represents the value of the LTP weight in accordance with the color map in Fig. 3f. A post-synaptic neuron, j, that is connected with a red arrow from a pre-synaptic neuron, i, is more likely to spike than others, if the other conditions are equivalent, shortly after the spike from the pre-synaptic neuron reaches the post-synaptic neuron after the conduction delay $d_{i,j}$.

Before training began (see Fig. 3a), the learnable parameters were independent of the target sequence and sampled independently from the normal distribution with mean 0.0 and standard deviation 0.1. The DyBM gradually learned appropriate values of the bias and the LTP weight as the training progresses (see Fig. 3b–e). It also learned the LTD weight (see Supplementary Fig. S3).

Next, we show that the trained DyBM can detect anomalies[41] in a sequence. We presented the sequence "SCIENSESCIENCE" to the DyBM that was trained to generate the sequence shown in Fig. 2e. This presented sequence has an anomaly in that the second "C" of the first instance of "SCIENCE" is replaced with "S." We did not train the DyBM but did update the eligibility traces and FIFO queues, while showing it the anomalous sequence.

Figure 4 shows that the DyBM detected the anomaly in the sequence. The lower part of the figure shows the anomalous sequence. The plot in the upper part of Fig. 4 shows the negative log-likelihood (score of anomaly), predicted by the DyBM, for each of the 7-bit patterns in the anomalous sequence. The higher the negative log-likelihood of a 7-bit pattern is, the less likely it is that the DyBM generates that 7-bit pattern, given the preceding patterns. The predicted negative log-likelihood increased by orders of magnitude when the DyBM was presented with the first 7-bit pattern of the anomalous "S." This 7-bit pattern itself is not an anomaly and appears elsewhere with a small negative log-likelihood. The negative

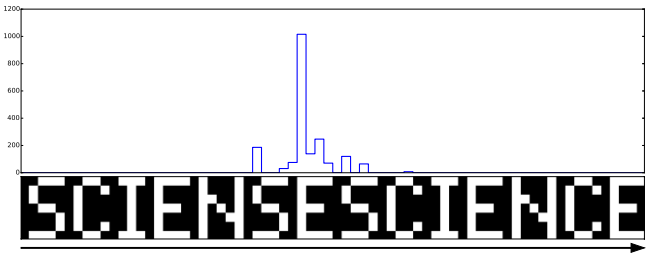


Figure 4. The DyBM trained with "SCIENCE" detected an anomaly in "SCIENSESCIENCE." After training the DyBM for 130,000 periods, we presented the DyBM with the anomalous sequence shown at the bottom of the figure. The top part shows the negative log-likelihood, predicted by the DyBM, for each of the seven-bit patterns in the anomalous sequence.

log-likelihood was indeed predicted to be small when the anomalous "S" is replaced with the normal "C." The initial patterns of the second "SCIEN" were predicted to have a high negative log-likelihood because they anomalously follow "SE."

Finally, we show that the DyBM can store multiple sequences and retrieve a particular sequence when an associated cue is presented to it. Here, we used two sequences, forward and reverse, from the alphabetical image "SCIENCE." The forward sequence was "SCIENCE," as in the previous experiments. The reverse sequence was created by reversing the order of the 7-bit patterns in the forward sequence, and thus is a sequential pattern from the mirror image of "SCIENCE." We trained the DyBM by presenting it with one of the two sequences in each iteration until it correctly retrieved that sequence when a partial sequence was presented as a cue. The cue for the forward sequence (forward cue) was the sequential pattern "SCIEN." The cue for the reverse sequence (reverse cue) was the sequential pattern of the mirror image of "IENCE."

Figure 5 shows how the DyBM learned the target sequences as the training progresses. The left images are the cues presented to the DyBM, and the right images are the sequential patterns that the DyBM generated after the corresponding cues were presented. We did not train the DyBM but did update its eligibility traces and FIFO queues when it was presented with cues or when it was generating sequential patterns. Here, the values of the eligibility traces and the FIFO queues were reset to zero before a cue was presented. This corresponds to presenting a sequence of zeros or a sequential pattern of a blank image to the DyBM for a sufficiently long period. Before training began (Fig. 5a), the DyBM generated neither of the target sequences, because the initial values of its learnable parameters were independent of the target sequences.

In the first iteration of training, we kept presenting the forward sequence to the DyBM. After the first iteration, the DyBM generated the forward sequence when the forward cue was presented (the upper part of Fig. 5b). At this point, the DyBM has not learned the reverse sequence. When the DyBM was presented with the reverse cue, it generated a sequence that starts with a part of "E" (the lower part of Fig. 5b). This happened because "E" follows "I" in the forward sequence, and the reverse cue ends with the mirror image of "I," which has vertical symmetry.

In the second iteration, we kept presenting the reverse sequence to the DyBM. After the second iteration, the DyBM generated the reverse sequence when the reverse cue was presented (the lower part of Fig. 5c). However, this DyBM lost its memory about the forward sequence and did not retrieve it when the forward cue was presented (the upper part of Fig. 5c). After the second iteration, we presented the forward sequence to the DyBM in odd iterations and the reverse sequence in even iterations. Figure 5d shows the sequences generated by the DyBM after one million iterations. The DyBM generated the reverse sequence, which was presented in the last iteration of training, but not the forward sequence.

Training the DyBM completed in 223 minutes after 1,785,845 iterations, at which point the DyBM learned the forward sequence without losing its memory of the reverse sequence learned in the preceding iteration. Figure 5e shows that the completely trained DyBM correctly generated the target sequences in accordance with the cues. Figure 6 shows the number of periods needed in each iteration of the training in Fig. 5. The first iteration finished in 120,701 periods. Each iteration took at most 5,837 periods after 1,000 iterations and at most 233 periods after 1,000,000 iterations. The DyBM learned a sequence quickly if it had already learned that sequence and forgotten it.

Additional results

The DyBM is limited neither to seven neurons nor to alphabetical images. Here, we apply the DyBM to additional two sequences. We use varying number of neurons, depending on the dimension of the bit-patterns. First, we trained the DyBM with the sequence that represents the motion picture of human evolution, illustrated in Supplementary Fig. S1f. Here, we used 20 neurons in the DyBM to learn this sequence of 20-bit patterns having the period of 41 units of time. Training completed in 19 seconds. Supplementary Fig. S1 shows the sequences that were generated by the trained DyBM. Second, we trained

(a) Before training

(b) After one iteration of training

(c) After 2 iterations of training

(d) After 1,000,000 iterations of training

(e) After 1,785,845 iterations of training

Figure 5. The DyBM learned two target sequences and retrieved a particular sequence when an associated cue was presented. The DyBM generated the right sequence after the left sequence was presented as a cue. (**a**) The DyBM did not generate the target sequences prior to training. (**b**) The DyBM learned the first target sequence "SCIENCE" in the first iteration and retrieved it when its partial sequence was presented as a cue. (**c**) After the second iteration, the DyBM retrieved the second target sequence of the mirror image of "SCIENCE" but not the first. (**d**) After one million iterations, the DyBM learned the second target sequence, while keeping a partial memory of the first. (**e**) The DyBM eventually retrieved one of the two target sequences, depending on the cue.

the DyBM with the sequence that represents a music, a simplified version of *Ich bin ein Musikante*, illustrated in Supplementary Fig. S2f. Here, we used 12 neurons in the DyBM, corresponding to 12 distinct notes. Training completed in 28 minutes. Supplementary Fig. S2 shows the sequences that were generated by the trained DyBM.

Discussion

The DyBM is trained by incrementally increasing the likelihood of the given dynamic patterns according to a learning rule that exhibits some of the properties of STDP. This training is analogous to training a conventional Boltzmann machine according to the Hebb rule such that the likelihood of the given static patterns is incrementally increased. Unlike the learning rules for the existing models that extend Boltzmann machines to deal with sequential patterns[32,42–45], our learning rule exactly increases the likelihood without approximations. The recent success of artificial neural networks in a number of engineering applications suggests that exact learning will be very useful[46,47]. The sequence of 7-bit patterns of

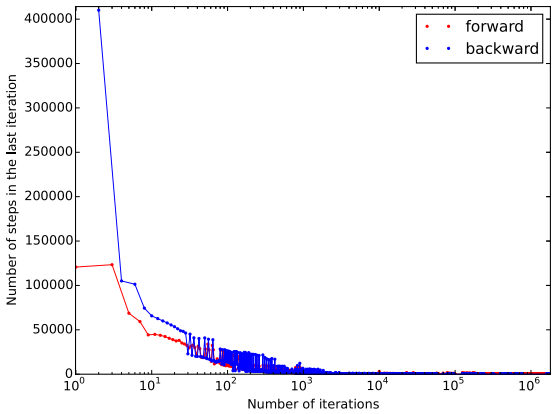


Figure 6. The DyBM quickly learned or recalled a target sequence that it had forgotten in previous iterations. The figure plots the number of periods that the DyBM took to learn a target sequence in each iteration of training in Fig. 5 against the number of iterations that have been completed.

the alphabetical image "SCIENCE" has a period of 35, which is significantly longer than the conduction delay, which ranges between 1 and 9. Learning this sequence involves finding a complex interaction among the neurons, FIFO queues, and eligibility traces by adjusting the values of the learnable parameters. A slight deviation in the learnable parameters from the appropriate values results in a failure to generate the target sequences, as is evident in Fig. 2 and Fig. 3.

Each learnable parameter of a DyBM can be updated only with information that is locally available in space and time, which is an important aspect of a learning rule of biological neurons[48] and is also a desirable property when it comes to implementation as hardware. Specifically, the bias of a neuron is updated with the eligibility traces and FIFO queues associated with that neuron, its pre-synaptic neuron, or its post-synaptic neurons. The weight of a synapse is also updated with the information associated with its pre-synaptic neuron and its post-synaptic neuron. Although the DyBM learns the conditional probability distribution of the current patterns given the preceding patterns, it does not store all of the preceding patterns. Instead, the information about the preceding patterns is aggregated into the latest values of the eligibility traces and FIFO queues. The eligibility traces, either neural or synaptic, and the FIFO queues are also updated only with local information, *i.e.*, whether or not a spike is generated or has arrived at an associated neuron.

Similar to standard recurrent neural networks[40,49,50], the DyBM can be unfolded though time[51,52]. The unfolded DyBM is a Boltzmann machine having an infinite number of units, each representing the value of a neuron at a particular time. In particular, the units representing the most recent values of neurons are not connected to each other. The weights on connections share a finite number of values and are trained via discriminative learning[53].

To date, STDP had limited success in engineering applications primarily because of the lack of sufficient underpinnings. This situation is analogous to the history of the Hebb rule. Although the Hebb rule motivated early studies on artificial neural networks prior to the 1980's[14], it had limited success until it was given an underpinning in the form of the Hopfield network[10] and the Boltzmann machine[15–17]. The DyBM does the same for STDP. In fact, the learning rule of the DyBM also exhibits a form of homeostatic plasticity for keeping the spiking probability relatively constant[54,55]. The DyBM thus provides an underpinning for homeostatic STDP.

Methods

Here, we describe the details of the dynamic Boltzmann machine (DyBM), the learning rule for the DyBM, and the parameters of the DyBM used in the experiments.

Dynamic Boltzmann machine. A DyBM consists of a set of neurons having memory units and first-in-first-out (FIFO) queues. Let N be the number of neurons. Each neuron takes a binary value of either 0 or 1 at each moment. For $j \in [1, N]$, let $x_j^{[t]}$ be the value of the j-th neuron at time t.

A neuron, $i \in [1, N]$, may be connected to another neuron, $j \in [1, N]$, with a FIFO queue of length $d_{i,j} - 1$, where $d_{i,j}$ is the axonal or synaptic delay of conductance, or conduction delay, from pre-synaptic neuron i to post-synaptic neuron j. Note that any neuron can be called a pre-synaptic neuron and a post-synaptic neuron, depending on the synapse under consideration. We assume $d_{i,j} \geq 1$. At each moment t, the tail of the FIFO queue holds $x_i^{[t-1]}$, and the head of the FIFO queue holds $x_i^{\left[t-d_{i,j}+1\right]}$. As the time progresses by one unit, the value at the head of the FIFO queue is removed, the remaining values in the FIFO queues are pushed toward the head by one position, and a new value is inserted at the tail of the FIFO queue. We allow a neuron to be connected to itself via a FIFO queue.

Each neuron stores a fixed number, L, of neural eligibility traces. For $\ell \in [1, L]$ and $j \in [1, N]$, let $\gamma_{j,\ell}^{[t-1]}$ be the ℓ-th neural eligibility trace of the j-th neuron immediately before time t:

$$\gamma_{j,\ell}^{[t-1]} \equiv \sum_{s=-\infty}^{t-1} \mu_\ell^{\,t-s} x_j^{[s]}, \tag{1}$$

where $\mu_\ell \in (0, 1)$ is the decay rate for the ℓ-th neural eligibility trace. That is, the neural eligibility trace is the weighted sum of the past values of that neuron, where the recent values have greater weights than older ones.

Each neuron also stores synaptic eligibility traces, where the number of the synaptic eligibility traces depends on the number of the neurons that are connected to that neuron. Namely, for each of the (pre-synaptic) neurons that are connected to a (post-synaptic) neuron j, the neuron j stores a fixed number, K, of synaptic eligibility traces. For $k \in [1, K]$, let $\alpha_{i,j,k}^{[t-1]}$ be the k-th synaptic eligibility trace of neuron j for pre-synaptic neuron i immediately before time t:

$$\alpha_{i,j,k}^{[t-1]} \equiv \sum_{s=-\infty}^{t-d_{i,j}} \lambda_k^{\,t-s-d_{i,j}} x_i^{[s]}, \tag{2}$$

where $\lambda_k \in (0, 1)$ is the decay rate for the k-th synaptic eligibility traces. That is, the synaptic eligibility trace is the weighted sum of the values that has reached that neuron, j, from a pre-synaptic neuron, i, after the conduction delay, $d_{i,j}$. Again, the recent values have greater weights than older ones.

The values of the eligibility traces stored at neuron j are updated locally at time t using the value of neuron j at time t and the values that have reached neuron j at time t from its pre-synaptic neurons. Specifically,

$$\gamma_{j,\ell}^{[t]} \leftarrow \mu_\ell \left(\gamma_{j,\ell}^{[t-1]} + x_j^{[t]} \right) \tag{3}$$

$$\alpha_{i,j,k}^{[t]} \leftarrow \lambda_k \left(\alpha_{i,j,k}^{[t-1]} + x_i^{t-d_{i,j}} \right) \tag{4}$$

for $\ell \in [1, L]$ and $k \in [1, K]$, and for neurons i that are connected to j.

The DyBM has learnable parameters that are updated during training, in addition to the structural parameters (Supplementary Table S1a) and variables (Supplementary Table S1c), which have been introduced in the preceding. The learnable parameters of the DyBM are the bias and weight (Supplementary Table S1b). Specifically, each neuron, j, is associated with a bias, b_j. Each synapse, or each pair of neurons that are connected via a FIFO queue, is associated with the weight of long term potentiation (LTP weight) and the weight of long term depression (LTD weight). The LTP weight from a (pre-synaptic) neuron, i, to a (post-synaptic) neuron, j, is characterized with K parameters, $u_{i,j,k}$ for $k \in [1, K]$. The k-th LTP weight corresponds to the k-th synaptic eligibility trace for $k \in [1, K]$. The LTD weight from a (pre-synaptic) neuron, i, to a (post-synaptic) neuron, j, is characterized with L parameters, $v_{i,j,\ell}$ for $\ell \in [1, L]$. The ℓ-th LTD weight corresponds to the ℓ-th neural eligibility trace for $\ell \in [1, L]$. The learnable parameters are collectively denoted as θ.

We now define the energy of the DyBM, using the notations introduced in the preceding. Similar to the conventional Boltzmann machine[15–17], the energy of the DyBM determines what patterns of values that the DyBM is more likely to generate. Contrary to the conventional Boltzmann machine, the energy associated with a pattern at a moment depends on the patterns that the DyBM has previously generated. Let $\mathbf{x}^{[t]} = \left(x_j^{[t]}\right)_{j\in[1,N]}$ be the vector of the values of the neurons at time t. Let $\mathbf{x}^{[:t-1]} = (\mathbf{x}^{[s]})_{s<t}$ be the sequence of the values of the DyBM before time t. The energy of the DyBM at time t depends not only on $\mathbf{x}^{[t]}$, but also on $\mathbf{x}^{[:t-1]}$, which is stored as eligibility traces in the DyBM. Let $E_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[:t-1]})$ be the energy of the DyBM at time t. The lower the energy of the DyBM with particular values $\mathbf{x}^{[t]}$, the more likely the DyBM takes those values.

The energy of the DyBM can be decomposed into the energies of the individual neurons at time t:

$$E_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[:t-1]}) = \sum_{j=1}^{N} E_\theta\left(x_j^{[t]}\middle|\mathbf{x}^{[:t-1]}\right). \tag{5}$$

The energy of neuron j at time t depends on the value it takes as follows:

$$E_{\theta,j}\left(x_j^{[t]}\middle|\mathbf{x}^{[:t-1]}\right) = -b_j x_j^{[t]} - \sum_{i=1}^{N}\sum_{k=1}^{K} u_{i,j,k}\alpha_{i,j,k}^{[t-1]} x_j^{[t]} + \sum_{i=1}^{N}\sum_{\ell=1}^{L} v_{i,j,\ell}\beta_{i,j,\ell}^{[t-1]} x_j^{[t]} + \sum_{i=1}^{N}\sum_{\ell=1}^{L} v_{j,i,\ell}\gamma_{i,\ell}^{[t-1]} x_j^{[t]}, \tag{6}$$

where we define

$$\beta_{i,j,\ell}^{[t-1]} \equiv \sum_{s=t-d_{i,j}+1}^{t-1} \mu_\ell^{s-t} x_i^{[s]}. \tag{7}$$

The first term of the right side of equation (6) shows that, roughly speaking, a neuron having a large positive bias is likely to spike ($x_j^{[t]} = 1$) at any time t, because its energy tends to be low when it spikes. More precisely, the energy of the neuron is determined by the balance among the four terms on the right side of equation (6).

The second term of the right side corresponds to LTP. Consider a pair of a pre-synaptic neuron, i, and a post-synaptic neuron, j, whose LTP weight, $u_{i,j,k}$ for $k \in [1, K]$, has a large positive value. Then j is likely to spike at time t, if the spikes from i have arrived shortly before time t, which makes $\alpha_{i,j,k}^{[t-1]}$ large for $k \in [1, K]$.

The third term of the right side corresponds to LTD. LTD suggests that a post-synaptic neuron, j, is unlikely to spike shortly before a spike from a pre-synaptic neuron, i, reaches j. The corresponding LTD weight, $v_{i,j,\ell}$ for $\ell \in [1, L]$, controls the strength of LTD for that synapse. The FIFO queue holds the spikes generated by i for the last $d_{i,j} - 1$ units of time, and those spikes reach j within a short period of at most $d_{i,j} - 1$ units of time. Here, $\beta_{i,j,\ell}^{[t-1]}$, as defined in equation (7), takes a positive value when the FIFO queue from neuron i to neuron j has spikes ($x_i^{[s]} = 1$ for some of $s \in [t - d_{i,j} + 1, t - 1]$). The more spikes the FIFO queue has, and the closer to the head of the FIFO queue those spikes are, the larger the value of $\beta_{i,j,\ell}^{[t-1]}$ will be. Equation (6) suggests that the post-synaptic neuron, j, is unlikely to spike, when $\beta_{i,j,\ell}^{[t-1]}$ is large and the corresponding LTD weight, $v_{i,j,\ell}$ for $\ell \in [1, L]$, has a large positive value.

The last term of the right side can also be considered LTD. The intuition is that the third term only considers the spikes that are going to reach the post-synaptic neuron within the period of conduction delay, while the last term considers the spikes that are going to arrive after the conduction delay. In the last term, neuron i plays the role of a post-synaptic neuron, and neuron j is a pre-synaptic neuron, contrary to their roles in the third term. The LTD weight is thus $v_{j,i,\ell}$, rather than $v_{i,j,\ell}$, for $\ell \in [1, L]$.

Consider a synapse from a pre-synaptic neuron, j, to a post-synaptic neuron, i, that has a large positive value of $v_{j,i,\ell}$ for $\ell \in [1, L]$. The neural eligibility trace, $\gamma_{i,\ell}^{[t-1]}$, stores information about the spikes that the post-synaptic neuron i generated before time t. Namely, the value of $\gamma_{i,\ell}^{[t-1]}$ is high when neuron i spikes shortly before time t. The post-synaptic neuron is unlikely to have spiked shortly before time t, if the pre-synaptic neuron spikes at time t, which will reach the post-synaptic neuron after the conduction delay. Specifically, if i spiked at time $t - \delta$ and j spiked at time t, then that spike from j would reach i after the conduction delay, at time $t + d_{j,i}$, which is $\delta + d_{j,i}$ units of time after i spiked. LTD suggests that such a pair of spikes is unlikely to be generated when $\delta + d_{j,i}$ is small. The causality is, however, that the pre-synaptic neuron is unlikely to spike at time t (*i.e.*, $x_j^{[t]} = 1$) if the post-synaptic neuron has recently spiked.

The probability that a neuron, j, takes a particular value, $x_j^{[t]} \in \{0, 1\}$, at time t depends on $\mathbf{x}^{[:t-1]}$, through the values of the eligibility traces, and can be expressed as follows:

$$P_{\theta,j}\left(x_j^{[t]} \middle| \mathbf{x}^{[:t-1]}\right) = \frac{\exp\left(-\tau^{-1} E_{\theta,j}\left(x_j^{[t]} \middle| \mathbf{x}^{[:t-1]}\right)\right)}{\sum_{\tilde{x} \in \{0,1\}} \exp\left(-\tau^{-1} E_{\theta,j}(\tilde{x} | \mathbf{x}^{[:t-1]})\right)}, \tag{8}$$

where τ is a parameter called temperature. The values of the neurons at time t are conditionally independent of each other, given $\mathbf{x}^{[:t-1]}$. Then the probability that the DyBM takes $\mathbf{x}^{[t]}$ at time t can be expressed as follows:

$$P_\theta(\mathbf{x}^{[t]} | \mathbf{x}^{[:t-1]}) = \prod_{j=1}^{N} P_{\theta,j}\left(x_j^{[t]} \middle| \mathbf{x}^{[:t-1]}\right). \tag{9}$$

The corresponding log-likelihood is given by

$$\log\ P_\theta(\mathbf{x}^{[t]} | \mathbf{x}^{[:t-1]}) = \sum_{j=1}^{N} \log\ P_{\theta,j}\left(x_j^{[t]} \middle| \mathbf{x}^{[:t-1]}\right). \tag{10}$$

We generated the sequential patterns in Fig. 2 and Fig. 5 by using equation (8) with an infinitesimally small temperature. Specifically, we let $\tau \to 0$, so that equation (8) only takes 0 or 1. At each moment t, the values of the neurons, $x_j^{[t]}$ for $j \in [1, N]$, are deterministically generated. Those values are used to update the eligibility traces, before the values at the next moment, $t + 1$, are generated.

Figure 4 shows the magnitudes of the log-likelihood (or negative log-likelihood) calculated using equation (10). The values, $x_j^{[t]}$ for $j \in [1, N]$, presented to the DyBM to calculate its log-likelihood at

time t are used to update the eligibility traces, before the values at the next moment, $t+1$, are presented.

Learning rule. The probability distribution of the values that the DyBM generates depends on the values of the learnable parameters, θ, as is evident in equation (8). Below, we describe how to train those learnable parameters.

When the DyBM is presented with the values, $\mathbf{x}^{[t]}$, at time t, we update its learnable parameters in the direction of increasing log-likelihood:

$$\theta \leftarrow \theta + \eta \nabla_\theta \ \log \ P_\theta(\mathbf{x}^{[t]}|\mathbf{x}^{[:t-1]}), \tag{11}$$

where η is the learning rate. This in turn increases the log likelihood of the sequence of the values up to time t:

$$\log \ p\,(\mathbf{x}^{[:t]}) = \sum_{s\le t} \log \ P_\theta(\mathbf{x}^{[s]}|\mathbf{x}^{[:s-1]}). \tag{12}$$

During training, we keep the temperature at $\tau = 1$.

Specifically, when the value, $x_j^{[t]}$, is presented to a neuron, j, its bias is updated as follows:

$$b_j \leftarrow b_j + \eta \left(x_j^{[t]} - \left\langle X_j^{[t]} \right\rangle_\theta\right), \tag{13}$$

where

$$\left\langle X_j^{[t]} \right\rangle_\theta = P_{\theta j}(1|\mathbf{x}^{[:t-1]}) \tag{14}$$

denotes the expectation of the value that j generates at time t given the values of the parameters and variables of the DyBM immediately before that time. Namely, the bias is increased if the value presented to the neuron is greater than what is expected. Otherwise, the bias is decreased. The magnitude of the update is proportional to the difference between the presented value and the expected value. The learning rule for the bias is similar to the Hebb rule[16,17] except that, here, the expression of expectation (14) depends on the past values of the neurons via equation (8). The term with the expectation has the role of homeostatic plasticity, which keeps the spiking probability relatively constant[54,55]. Analogous terms of homeostatic plasticity will appear in the following learning rule for the other parameters.

The LTP weight is increased or decreased analogously to a bias as follows:

$$u_{i,j,k} \leftarrow u_{i,j,k} + \eta \left(x_j^{[t]} - \left\langle X_j^{[t]} \right\rangle_\theta\right) \alpha_{i,j,k}^{[t-1]} \tag{15}$$

for each $i, j \in [1, N]$ and $k \in [1, K]$. Now, the magnitude of the update is also proportional to the corresponding synaptic eligibility trace, $\alpha_{i,j,k}^{[t-1]}$. The learning rule for the LTP weight takes a form that is analogous to the REINFORCE algorithm[56], where η is called a learning rate factor, $x_j^{[t]}$ corresponds to reinforcement, $\left\langle X_j^{[t]} \right\rangle_\theta$ corresponds to the reinforcement baseline, and $\alpha_{i,j,k}^{[t-1]}$ is called a characteristic eligibility, although the particular forms of individual factors are unique to the DyBM.

We can also express equation (15) as follows:

$$u_{i,j,k} \leftarrow u_{i,j,k} + \eta \left(\alpha_{i,j,k}^{[t-1]} x_j^{[t]} - \left\langle \alpha_{i,j,k}^{[t-1]} X_j^{[t]} \right\rangle_\theta\right) \tag{16}$$

Namely, the LTP weight is increased if the product of the presented value and the value of the synaptic eligibility trace is greater than what is expected. Otherwise, the LTP weight is decreased. Increasing $u_{i,j,k}$ results in increasing the probability that neuron j spikes particularly at the time s when $\alpha_{i,j,k}^{[s]}$ is high. This is what we expect with LTP.

The LTD weight is increased or decreased, depending on two terms:

$$v_{i,j,\ell} \leftarrow v_{i,j,\ell} + \eta \left(\left\langle X_j^{[t]} \right\rangle_\theta - x_j^{[t]}\right) \beta_{i,j,\ell}^{[t-1]} + \eta \left(\langle X_i^{[t]} \rangle_\theta - x_i^{[t]}\right) \gamma_{j,\ell}^{[t-1]} \tag{17}$$

for each $i, j \in [1, N]$, and $\ell \in [1, L]$. Again, we can express equation (17) equivalently as follows:

$$v_{i,j,\ell} \leftarrow v_{i,j,\ell} + \eta \left(\left\langle \beta_{i,j,\ell}^{[t-1]} X_j^{[t]} \right\rangle_\theta - \beta_{i,j,\ell}^{[t-1]} x_j^{[t]}\right) + \eta \left(\left\langle \gamma_{j,\ell}^{[t-1]} X_i^{[t]} \right\rangle_\theta - \gamma_{j,\ell}^{[t-1]} x_i^{[t]}\right), \tag{18}$$

which can then be divided into two steps:

$$\nu_{i,j,\ell} \leftarrow \nu_{i,j,\ell} + \eta \left(\left\langle \beta_{i,j,\ell}^{[t-1]} X_j^{[t]} \right\rangle_\theta - \beta_{i,j,\ell}^{[t-1]} x_j^{[t]} \right) \tag{19}$$

$$\nu_{i,j,\ell} \leftarrow \nu_{i,j,\ell} + \eta \left(\left\langle \gamma_{j,\ell}^{[t-1]} X_i^{[t]} \right\rangle_\theta - \gamma_{j,\ell}^{[t-1]} x_i^{[t]} \right). \tag{20}$$

Equation (19) involves the product of $\beta_{i,j,\ell}^{[t-1]}$ and $x_j^{[t]}$, where we should recall that the value of $\beta_{i,j,\ell}^{[t-1]}$ in equation (7) depends on the spikes traveling from neuron i to neuron j. The LTD weight is increased if the expected value of this product is greater than the corresponding observed value. Increasing $\nu_{i,j,l}$ decreases the probability that neuron j spikes at time t when $\beta_{i,j,\ell}^{[t-1]}$ is high, which is what we expect with LTD. Equation (20) involves the product of $\gamma_{j,\ell}^{[t-1]}$, the neural eligibility trace of neuron j, and $x_i^{[t]}$, the value of neuron i. The LTD weight is increased if the expected value of this product is greater than the corresponding observed value. Increasing $\nu_{i,j,l}$ decreases the probability that neuron i spikes at time s when $\gamma_{j,\ell}^{[s]}$ is high.

We trained the DyBM with the online gradient ascent method[33] before letting it generate the sequential patterns shown in Fig. 2 and Fig. 5 and before presenting it with the anomalous sequence shown in Fig. 4. Specifically, after presenting the value, $\mathbf{x}^{[t]}$, to the DyBM at time t, we update its bias according to equation (13), its LTP weight according to equation (15), and its LTD weight according to equations (19 and 20). Before presenting the next value, $\mathbf{x}^{[t+1]}$, to the DyBM, we also update the eligibility traces according to equations (1 and 2) as well as the FIFO queues.

We initially set the learning rate at $\eta = 1$ and adjust it for each parameter as the training progresses, using AdaGrad[34]. Specifically, for $m \geq 0$, let η_m be the learning rate of a parameter, $\xi \in \theta$, after ξ is updated m times. We update ξ according to

$$\xi \leftarrow \xi + \eta_m \Delta_m, \tag{21}$$

where Δ_m is the corresponding derivative of the log-likelihood, equation (10), with respect to ξ. For $m \geq 1$, we adjust η_m according to

$$\eta_m = \frac{\eta_0}{\sqrt{\sum_{s=0}^{m-1} \Delta_s^2}}, \tag{22}$$

where we set $\eta_0 = 1$. More precisely, the learning rates for the LTD weight are adjusted independently between equation (19) and equation (20) for each $i, j \in [1, N]$ and $\ell \in [1, L]$.

Parameters. Throughout the experiments, the N neurons were densely connected to each other, and the conduction delay was sampled from an integer uniform distribution between 1 and 9 (see Supplementary Table S2). In particular, each neuron was connected to itself via a FIFO queue. Each neuron held $L = 3$ neural eligibility traces, whose decay rates were $\mu_1 = 0.25$, $\mu_2 = 0.5$, or $\mu_3 = 0.75$. A neuron also held $K = 3$ synaptic eligibility traces, whose decay rates were $\lambda_1 = 0.25$, $\lambda_2 = 0.5$, or $\lambda_3 = 0.75$, for each of the N FIFO queues coming into that neuron. The values of these structural parameters were fixed.

Before training, we set the initial values of the variables (Supplementary Table S1c) to 0. This corresponds to presenting a sequence of blank patterns, or zero vectors, for a sufficiently long period before the training. We sample the initial values of the learnable parameters (Supplementary Table S1b) independently from a normal distribution with mean 0.0 and standard deviation 0.1.

We implemented the algorithm for training DyBMs in Java™ and executed it on a Java Virtual Machine with the default setting for the maximum heap size (2 GB) and ran it on a single thread of an Intel Xeon E5-2670 processor on a workstation equipped with Microsoft Windows 7 Professional edition.

References

1. Ciresan, D., Meier, U., Masci, J. & Schmidhuber, J. A committee of neural networks for traffic sign classification. In *Proc. 2011 International Joint Conference on Neural Networks (IJCNN)*, 1918–1921, San Jose, CA, USA. New York, NY, USA: IEEE (2011, July 31 - August 5).
2. Yu, D. & Seltzer, M. L. Improved bottleneck features using pretrained deep neural networks. In *Proc. 12th Annual Conference of the International Speech Communication Association (INTERSPEECH 2011)*, 237–240, Florence, Italy. Red Hook, NY, USA: Curran Associates, Inc. (2011, August 27-31).
3. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012*, 1106–1114, Lake Tahoe, NV, USA (Curran Associates, Inc., Red Hook, NY, USA, 2012, December 3-8).
4. Ivakhnenko, A. G. & Lapa, L. G. *Cybernetic Predicting Devices* (CCM Information Corporation, 1965).
5. Ballard, D. H. Modular learning in neural networks. In *Proc. 6th National Conference on Artificial Intelligence (AAAI-87)*, 279–284, Seattle, WA, USA. Burlington, MA, USA: Morgan Kaufmann (1987, July 13-17).
6. Hinton, G. E. & Salakhutdinov, R. Reducing the dimensionality of data with neural networks. *Science* **313,** 504–507 (2006).
7. Cireşan, D. C., Meier, U., Gambardella, L. M. & Schmidhuber, J. Deep, big, simple neural nets for handwritten digit recognition. *Neural Comput.* **22,** 3207–3220 (2010).
8. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks* **61,** 85–117 (2015).

9. Kohonen, T. *Associative memory – Asystem-theoretic approach* (Springer, 1977).
10. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* **79,** 2554–2558 (1982).
11. Gerstner, W. & van Hemmen, J. L. Associative memory in a network of 'spiking' neurons. *Network* **3,** 139–164 (1992).
12. Gerstner, W., Ritz, R. & van Hemmen, J. L. Why spikes? Hebbian learning and retrieval of time-resolved excitation patterns. *Biol. Cybern.* **69,** 503–515 (1993).
13. Hebb, D. O. *The organization of behavior: A neuropsychological approach* (Wiley, 1949).
14. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* **65,** 386–408 (1958).
15. Fahlman, S. E., Hinton, G. E. & Sejnowski, T. J. Massively parallel architectures for AI: NETL, Thistle, and Boltzmann machines. In *Proc. National Conference on Artificial Intelligence (AAAI-83)*, 109–113, Washington, D.C., USA. Menlo Park, CA, USA: AAAI Press (1983, August 22–26).
16. Hinton, G. E. & Sejnowski, T. J. Optimal perceptual inference. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 448–453, Washington D.C., USA. New York, NY, USA: IEEE (1983, June 19-23).
17. Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. A learning algorithm for Boltzmann machines. *Cognitive Science* **9,** 147–169 (1985).
18. Gerstner, W., Kempter, R., van Hemmen, J. L. & Wagner, H. A neuronal learning rule for sub-millisecond temporal coding. *Nature* **383,** 76–78 (1996).
19. Abbott, L. F. & Nelson, S. B. Synaptic plasticity: Taming the beast. *Nature Neurosci.* **3,** 1178–1183 (2000).
20. Caporale, N. & Dan, Y. Spike timing-dependent plasticity: A Hebbian learning rule. *Annu. Rev. Neurosci.* **31,** 25–46 (2008).
21. Pachitariu, M. & Sahani, M. Learning visual motion in recurrent neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012*, 1322–1330, Lake Tahoe, NV, USA (Curran Associates, Inc., Red Hook, NY, USA, 2012, December 3-8).
22. Bliss, T. V. P. & Lømo, T. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *J. Physiol. Lond.* **232,** 331–356 (1973).
23. Bliss, T. V. P. & Gardner-Medwin, A. R. Long-lasting potentiation of synaptic transmission in the dentate area of the unanaestetized rabbit following stimulation of the perforant path. *J. Physiol. Lond.* **232,** 357–374 (1973).
24. Markram, H., Lübke, J., Frotscher, M. & Sakmann, B. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science* **275,** 213–215 (1997).
25. Bi, G. & Poo, M. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci.* **18,** 10464–10472 (1998).
26. Sjöström, P. J., Turrigiano, G. G. & Nelson, S. B. Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron* **32,** 1149–1164 (2001).
27. Klopf, A. H. *The hedonistic neuron: A theory of memory, learning, and intelligence* (Hemisphere, 1982).
28. Barto, A. G., Sutton, R. S. & Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst., Man, and Cybern.* **SMC-13,** 834–846 (1983).
29. Sutton, R. S. & Barto, A. G. Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Rev.* **88,** 135–170 (1981).
30. McCarthy, D. & Davison, M. Delayed reinforcement and delayed choice in symbolic matching to sample: Effects on stimulus discriminability. *J. Exp. Anal. Behav.* **46,** 293–303 (1986).
31. Killeen, P. R. Writing and overwriting short-term memory. *Psychon. Bull. Rev.* **8,** 18–43 (2001).
32. Sutskever, I., Hinton, G. E. & Taylor, G. W. The recurrent temporal restricted Boltzmann machine. In *Advances in Neural Information Processing Systems 21: 22th Annual Conference on Neural Information Processing Systems 2008*, 1601–1608, Vancouver, Canada (Curran Associates, Inc., Red Hook, NY, USA, 2008, December 8-11).
33. Bottou, L. Online algorithms and stochastic approximations. In *Online Learning and Neural Networks*, 9–42 (Cambridge University Press, Cambridge, UK, 1998).
34. Duchi, J., Hazan, E. & Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **12,** 2121–2159 (2011).
35. Kelley, H. J. Gradient theory of optimal flight paths. *ARS J.* **30,** 947–954 (1960).
36. Bryson, A. E. A gradient method for optimizing multi-stage allocation processes. In *Proc. Harvard University Symposium on Digital Computers and Their Applications*, 125–135, Brookline, MA, USA. Cambridge, MA, USA: Harvard University Press (1961, April 3-6).
37. Dreyfus, S. E. The numerical solution of variational problems. *J. Math. Anal. and Appl.* **5,** 30–45 (1962).
38. Linnainmaa, S. *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*. Master's thesis, Univ. Helsinki (1970).
39. Werbos, P. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. thesis, Harvard Univ. (1974).
40. Werbos, P. J. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks* **1,** 339–356 (1988).
41. Chandola, V., Banerjee, A. & Kumar, V. Anomaly detection: A survey. *ACM Comput. Surveys* **41,** Article 15 (2009).
42. Hinton, G. E. & Brown, A. D. Spiking Boltzmann machines. In *Advances in Neural Information Processing Systems 12: 13th Annual Conference on Neural Information Processing Systems 1999*, 122–128, Denver, CO, USA (The MIT Press, Cambridge, MA, USA, 1999, November 29 - December 4).
43. Sutskever, I. & Hinton, G. E. Learning multilevel distributed representations for high-dimensional sequences. *J. Mach. Learn. Res. W&P* **2,** 548–555 (2007).
44. Taylor, G. W. & Hinton, G. E. Factored conditional restricted Boltzmann machines for modeling motion style. In *Proc. 26th Annual International Conference on Machine Learning (ICML 2009)*, 1025–1032, Montreal, Canada. New York, NY, USA: ACM (2009, June 14-18).
45. Mittelman, R., Kuipers, B., Savarese, S. & Lee, H. Structured recurrent temporal restricted Boltzmann machines. In *Proc. 31st Annual International Conference on Machine Learning (ICML 2014)*, 1647–1655, Beijing, China. Brookline, MA, USA: Microtome Publishing (2014, June 21-26).
46. Graves, A. *et al.* A novel connectionist system for improved unconstrained handwriting recognition. *IEEE Trans. Pattern Anal.* **31,** 855–868 (2009).
47. Graves, A., Mohamed, A. & Hinton, G. Speech recognition with deep recurrent neural networks. In *Proc. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6645–6649, Vancouver, Canada. New York, NY, USA: IEEE (2013, May 26-31).
48. Gerstner, W. & Kistler, W. M. *Spiking Neuron Models: Single Neurons, Populations, Plasticity* (Cambridge University Press, 2002).
49. Elman, J. L. Finding structure in time. *Cognitive Science* **14,** 179–211 (1990).
50. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9,** 1735–1780 (1997).

51. Waibel, A., Hanazawa, T., Hinton, G., Shikano, K. & Lang, K. J. Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoust. Speech* **37,** 328–339 (1989).
52. Jaitly, N. & Hinton, G. Learning a better representation of speech soundwaves using restricted Boltzmann machines. In *Proc. 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5884–5887, Prague, Czech Republic. New York, NY, USA: IEEE (2011, May 22-27).
53. Larochelle, H. & Bengio, Y. Classification using discriminative restricted Boltzmann machines. In *Proc. 25th International Conference on Machine Learning (ICML 2008)*, 536–543, Helsinki, Finland. Madison, WI, USA: Omnipress (2008, July 5-9).
54. Turrigiano, G. G. & Nelson, S. B. Homeostatic plasticity in the developing nervous system. *Nature Rev. Neurosci.* **5,** 97–107 (2004).
55. Lazar, A., Pipa, G. & Triesch, J. SORN: A self-organizing recurrent neural network. *Frontiers in Computational Neurosci.* **3,** Article 23 (2009).
56. Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* **8,** 229–256 (1992).

Acknowledgments

This research was supported by CREST, JST.

Author Contributions

M.O. designed the DyBM and the learning rule without LTD, conducted preliminary experiments, suggested the directions of the experiments, and reviewed the manuscript. T.O. designed the parts of the DyBM and learning rule with LTD, implemented the learning algorithm, designed and conducted the experiments, and wrote the manuscript.

Additional Information

Supplementary information accompanies this paper at http://www.nature.com/srep

How to cite this article: Osogami, T. and Otsuka, M. Seven neurons memorizing sequences of alphabetical images via spike-timing dependent plasticity. *Sci. Rep.* **5**, 14149; doi: 10.1038/srep14149 (2015).