ReScience

[Re] Spike Timing Dependent Plasticity Finds the Start of Repeating Patterns in Continuous Spike Trains

Pamela Hathway[1] **and Dan F. M. Goodman,**[1]

1 Department of Electrical and Electronic Engineering, Imperial College, London, UK

p.hathway16@imperial.ac.uk

Editor
Name Surname

Reviewers
Name Surname
Name Surname

Received Sep, 1, 2015
Accepted Sep, 1, 2015
Published Sep, 1, 2015

Competing Interests:
The authors have declared that no competing interests exist.

Article repository

Code repository

A reference implementation of

→ Spike Timing Dependent Plasticity Finds the Start of Repeating Patterns in Continuous Spike Trains, Masquelier T, Guyonneau R, Thorpe SJ, PLoS ONE 3(1): e1377, 2008. https://doi.org/10.1371/journal.pone.0001377

Introduction

Neurons communicate through repeated, specifically timed action potential sequences (spike patterns) to convey information [2, 7]. Since neuronal activity is noisy and neurons are likely involved in a multitude of spike patterns of various lengths and extent, it can be hard to find spike patterns at first glance. The more neurons are recorded, the more difficult the task becomes due to the exponential increase of possible combinations of spikes that could make up a pattern [1]. It is unclear how neurons in the brain may extract relevant information from such input. In a 2008 paper, Masquelier and colleagues demonstrated that a single neuron with afferent synapses exhibiting spike timing dependent plasticity (STDP) is able to find the start of a repeating pattern in noisy (artificial) data [5].

Masquelier, Guyonneau, and Thorpe [5] use a simple feedforward network in which 2000 input neurons connect to one output neuron via excitatory STDP synapses. Half of the input neurons spike according to a spike pattern of 50 ms length for about 25% of the time. Finding the pattern is made more difficult by jittering the pattern spikes, adding random noise spikes to all neurons, and ensuring a constant population rate as well as no differences in overall firing rate between neurons.

We replicated their findings using the spiking neural network simulator Brian [4, 8], whereas the the original study implemented the simulations in Matlab, with the main functions being computed in C/C++ through mex files. In addition, we examined some of the implementation details and parameters and investigated whether they were essential to the success of the algorithm.

Methods

All simulations were performed using the spiking neural network simulator Brian (Brian2, version 2.0, http://briansimulator.org/). We attempted to stay as true as possible to the original study. Simulation parameters were taken from the text and we additionally obtained the source code for the standard parameter configuration from the authors to ensure equivalency of the implementations. The source code for deleting spikes within the pattern (Fig. 5 E) was not provided.

Running simulations

Brian calculates neuron properties such as membrane voltage at discrete time steps. The time step for these simulations was 10^{-4} s unless otherwise indicated. Each parameter combination was run 100 times and the success of a run determined as in the original study: a hit rate of over 98%, no false alarms and an average latency of under 10 ms (as calculated over the last 150 s of the simulation).

Table 1: Parameter variations

parameters	standard	variations
$w_{initial}$	0.475	0.275, 0.325, 0.375, 0.425
jitter (sd) [ms]	1	0, 2, 3, 4, 5, 6
pattern frequency	0.25	0.05, 0.1, 0.15, 0.5
prop. neurons in pattern	0.5	0.3, 0.4, 0.6
spike deletion	0	0.1, 0.2, 0.3

Spike trains

The spike trains of the 2000 input neurons were created in the same way as in the original study: a Poisson process with a variable instantaneous firing rate (30-90 Hz, 64 Hz on average) with no refractory period.

The absence of a refractory period in the input neurons has the effect that inter spike intervals can be as small as 10^{-9} s in some cases. The original implementation uses an event-based simulation method in which the simulation variables (output neuron voltage etc.) are calculated at every input neuron or output neuron spike, and so such small inter spike intervals are not a problem.

Brian on the other hand uses discrete time steps for its simulations and updates its simulation variables once per time step. In order to convert the original spike trains into Brian, we had to modify the spike trains slightly: whenever two spikes from the same neuron happened in the same time step, we deleted the second spike. At a resolution of 10^{-4} s (standard time step) this affected 0.25% of all spikes, and at a resolution of 10^{-6} s it affected 0.0025%. Deleting spikes does not affect the input drive to the neuron significantly since the initial firing rate of the output neuron is the same with or without the close spikes.

Leaky Integrate and Fire neuron

The original study models the potential of the output neuron with the Gerstner's Spike Response model (SRM) [3], which uses kernels to calculate the effect of incoming spikes on the postsynaptic voltage. Brian on the other hand uses differential equations to model the system parameters and evaluates those equations for each time step. We converted the kernels of the posystnaptic potential and the spike afterpotential into the following differential equations:

$$\frac{du}{dt} = \frac{Xx - u}{\tau_m} + \frac{Aa}{\tau_s} \tag{1}$$

$$\frac{dx}{dt} = -\frac{x}{\tau_{syn}} \tag{2}$$

$$\frac{da}{dt} = -\frac{A}{\tau_s} \tag{3}$$

The values for the parameters can be found in Tbl. 1. Eq. 1 describes the postsynaptic membrane potential, which is influenced by presynaptic excitatory postsynaptic

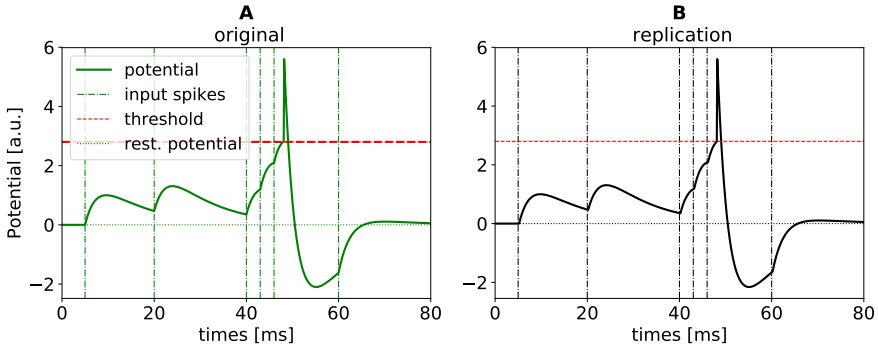

Figure 1: Postsynaptic potential. The postsynaptic potential of the replication implementation behaves the same as in the original study. Both show the same EPSP shapes and the negative afterpotential after a postsynaptic spike. **A**) The postsynaptic potential of the original study was calculated using the equations given in the original paper. **B**) The postsynaptic potential in the replication implementation was calculated from the differential equations specified in section **Leaky Integrate and Fire neuron**

potentials (EPSPs, first term) and a negative afterpotential after a spike (second term). In the event of a presynaptic spike, x is increased by 1 ($x \leftarrow x+1$), which initiates the voltage increase in the postsynaptic neuron. When the postsynaptic voltage reaches the threshold, a spike occurs: the voltage u is set to twice the threshold ($u \leftarrow 2T$), all EPSPs are flushed (all x set to 0, $x \leftarrow 0$) and the negative afterpotential is set into motion (a is set to 1, $a \leftarrow 1$). Presynaptic spikes had the same effect on the postsynaptic neuron potential as in the original publication as shown in Fig. 1.

Table 2: Simulation parameters

LIF neuron	value	.....	STDP	value
τ_m [ms]	10		τ_+ [ms]	16.8
τ_s [ms]	2.5		τ_- [ms]	33.7
τ_{syn} [ms]	2.5		a_+	2^{-5}
T [a.u.]	500		a_-	0.85 a_+
X	$\frac{\tau_s}{\tau_m}\frac{\tau_m}{\tau_s-\tau_m}$		w_{min}	0
A	-3 T		w_{max}	1
Δx	1		-	-
Δa	1		-	-

Spike Timing Dependent Plasticity

For synaptic plasticity, the original study uses the reduced nearest neighbor rule (RNN) [6] - a more restrictive version of the nearest neighbor (NN) STDP rule. In the standard NN rule, every spike causes a weight change (with the amount depending on the timing of the nearest spike), while for RNN a weight change only happens for the first postsynaptic spike immediately following a presynaptic spike (or vice versa). This means that for NN, there can be a series of potentiations or depressions, whereas for RNN potentiations and depressions strictly alternate.

A visualisation of three STDP rules is shown in Fig. 2: the RNN, the standard NN and the all-to-all (ATA) rule, in which all pairs of spikes are considered to calculate the weight change. A case is considered in which the presynaptic neuron (the input neurons in this study) spikes more often than the postsynaptic neuron (the output

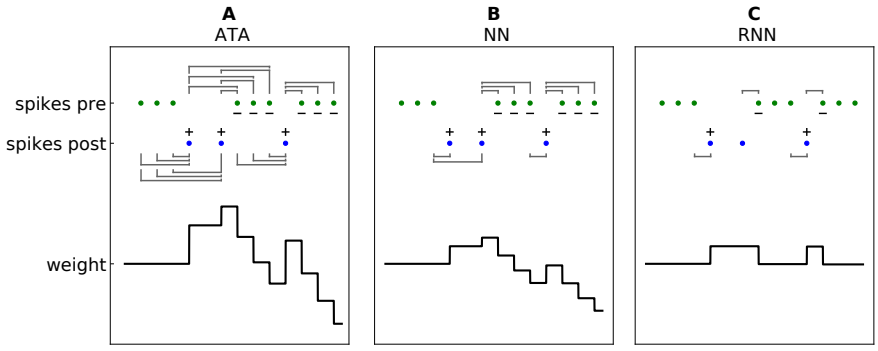

Figure 2: Effect of different learning rules on synaptic weight. We compare the all-to-all (ATA, **A**), nearest neighbor (NN, **B**), and reduced nearest neighbor (RNN, **C**) learning rules for a case in which the presynaptic neuron (green dots) spikes more often than the postsynaptic neuron (blue dots). Grey lines indicate which spike pairs are taken into account for the calculation of the synaptic weight change; the sign denotes a corresponding increase (+) or decrease (-) in weight at that spike. The synaptic weight in ATA (**A**) and NN (**B**) experience more weight changes and also a net decrease in weight after only a few spikes. Under the RNN rule (**C**), there are fewer changes and - in this example - no visible net weight change.

neuron in this study), leading to synaptic weight decreases for ATA and NN, but no visible net change under RNN (Fig. 2 **C**).

Results

The ability to find repeating patterns was reproduced in the replication implementation. We could qualitatively reproduce all results of the original paper and show the reproduced figures relevant to the model behaviour: latency development (Fig. 3), pattern specificity after convergence (Fig. 4), and robustness (Fig. 5). In addition, we comment on some implementation details that turned out to be relevant to replicating the original study successfully: simulation time step, learning rule, EPSP shape.

Pattern finding

The latency measures the time of the output neuron spike relative to the start of the pattern. If the spike occurs outside of the pattern (>50 ms after the start of the pattern), the latency for that spike is set to 0 as in the original paper.

In order to assess the time of pattern finding, we used the same spike train (standard conditions) as input for both the original code and in the replication implementation. The latency development looks similar in both implementations (see Fig. 3). The time until a stable state arises is longer in the replication implementation (1400 discharges instead of 700 or 20 s instead of 13.5 s). This discrepancy is due to the size of the discrete time steps used (see section **Implementation details**).

At the end of the simulation, the synaptic weights that are maximally potentiated (close to $w_{max} = 1$) belong exclusively to neurons involved in the pattern, whereas weights from neurons not involved in the pattern are depressed nearly completely (see Fig. 4). Neurons that spike at the beginning of the pattern are potentiated, causing the postsynaptic neuron to spike at a low latency.

Both latency development and the potentiation of only synaptic weights of pattern neurons in the converged state were successfully reproduced by the replication implementation.

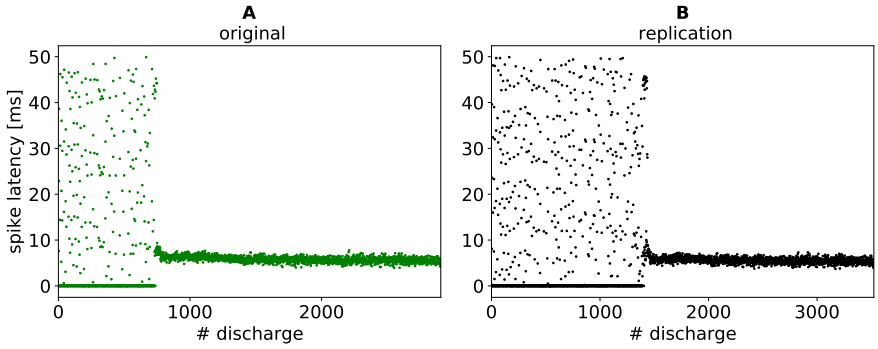

Figure 3: Latency. Comparing the latency development of the original study (**A**) and the replication implementation (**B**). In both implementations, the timing of the output spike in relation to the start of the pattern (vertical axis, set to latency=0 if output spike happens outside the pattern) is random at first, but becomes selective after a few hundred discharges. From then on the postsynaptic neuron always spikes only within the pattern (no spikes at latency = 0). The number of discharges until this happens is smaller in the original paper than in the replication implementation.

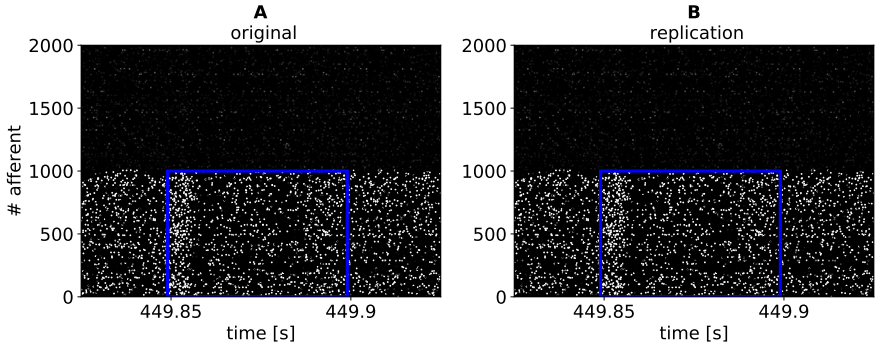

Figure 4: Weights at converged state. The weights in the converged state look very similar in the the original study (**A)**) and the replication implementation (**B)**). Weights from neurons not involved in the pattern are close to 0 (black, neurons 1000-1999), whereas weights from some neurons involved in the pattern (neurons 0-999) are close to 1 (white, maximum value). The weights of neurons that spike at the beginning of the pattern are nearly all close to 1 and their added postsynaptic potentials cause a spike at the beginning of the pattern (blue rectangle).

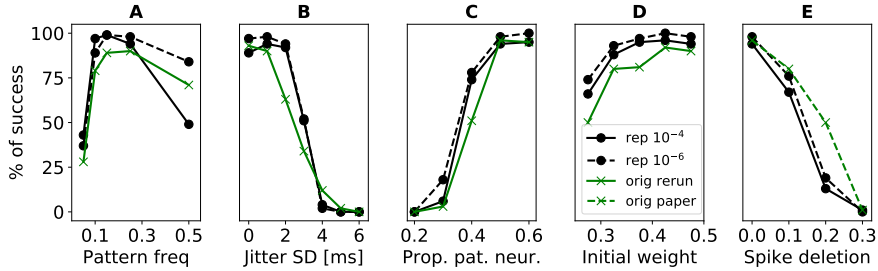

Figure 5: Robustness. Pattern finding abilities of the system were tested against different pattern presentation frequencies (**A**), jitter (**B**), number of neurons spiking according to the pattern (**C**), initial weights (**D**) and deletion of spikes in the pattern (**E**). For each parameter combination 100 simulations were run and the number of successful runs are reported as percentage of success. The replication implementation was run at two different time steps: 10^{-4} s (solid black lines) and 10^{-6} s (dashed black lines). The results from the original study were calculated using the source code provided (solid green lines in **A**, **B**, **C**, **D**) or estimated from the original publication (dashed green line in **E**).

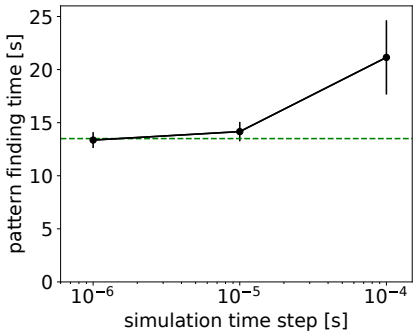

Figure 6: Time until finding pattern. Using larger simulation time steps leads to the pattern to be found later. The horizontal striped green line is the reported time when the pattern is found by Masquelier, Guyonneau, and Thorpe [5]. Errorbars represent standard deviation from 100 successful runs.

Robustness

Our simulations showed largely the same resilience to degradations as in the original paper, but despite the very similar implementations, there were some differences.

The replication implementation behaves the same as the original study when subjected to different amounts of jitter (Fig. 5 **B**), various proportions of neurons involved in the pattern (Fig. 5 **C**), different initial weights (Fig. 5 **D**) and when a percentage of spikes within the pattern are deleted (Fig. 5 **E**).

At a high pattern repetition frequency of 0.5 (Fig. 5 **A**), when the pattern is presented every 100 ms for 50 ms, the performance of the replication differs from the original: at larger time steps of 10^{-4} s the replication version does not perform as well as the original, but shows good results at smaller time steps (10^{-6} s).

Implementation details

We noticed that small implementation details can affect the behaviour of the network significantly. We summarised the relevant details in Table tbl. 3.

Simulation time step

In the standard parameter configuration, the time step chosen does not have an effect on overall pattern finding success, as long as the time step is less than or equal to 10^{-4} s. At larger time steps (10^{-3} s), no specificity emerges. Instead there is a systematic potentiation of all synaptic weights leading to very high firing rates in the output neuron. In contrast, success rates are above 95% for 10^{-4} s, 10^{-5} s, 10^{-6} s, and 10^{-7} s.

Although the success rate stays roughly the same, the time until the pattern is found (defined as the time after which no output spikes happen outside of the pattern) increases with larger time step size. An example can be seen in Fig. 3: the pattern is found after about 700 output neuron spikes in the original publication (left) and after about 1400 in the replication implementation (right) at a time step of 10^{-4} s for the same input. At smaller times steps (10^{-6} s), the pattern is found after about 700 discharges - the same as in the original publication (Fig. 6).

Table 3: Implementation details

	works	does not work
simulation time step	continuous, discrete	
time step size	$\leq 10^{-4}$ s	$> 10^{-4}$ s
learning rule	RNN	ATA, NN
EPSP shape	kernel	immediate voltage increase

Version of STDP learning rule

The choice of learning rule is crucial for the pattern finding abilities of the network. The RNN rule results in stable posystnaptic firing and reliable finding of the pattern. The use of other learning rules does not result in stable learning.

The spike trains used here involve neurons with a continuously high firing rate in the input neurons (on average 64 Hz, min 30 Hz, max 90 Hz) and a significantly lower firing rate in the output neuron (63 Hz initially, 5Hz after reaching specificity). This means that input neurons fire often in the time between output neuron spikes. Therefore, with any other than the reduced NN learning rule, the input neurons will experience a decrease in weight a lot more often (every time an input neuron spikes) than an increase in weight (every time the postsynaptic neuron spikes). This leads to a strong overall depression of the synaptic weights in the first few seconds of the simulation. With the parameters specified in Masquelier, Guyonneau, and Thorpe [5] and an ATA or a conventional NN learning rule, the output neuron stops firing after a few seconds because the output neuron voltage does not reach the voltage threshold necessary to evoke an output neuron spike anymore (see Fig. 7). In the case of the standard parameters, the output neuron stops firing after less than one second. When no output neuron spikes occur, learning cannot take place and no specificity can emerge.

In the case of an ATA learning rule, a behaviour resembling pattern finding can be evoked under the right circumstances: reducing the learning rate by a factor of 5 and increasing the value of a_+ (the maximum weight increase) to double the value of a_- (the maximum weight decrease). This setup works because the large amount of depression (on every input spike) is counteracted by large potentiation (due to the increased a_+). In such runs, the output neuron will correctly reach specificity and trace back through the latency, but then starts firing outside of the pattern again. This system is not stable since the ATA rule leads to "too much learning" as the synaptic weights change after every single spike. A further reduction in the learning rule will not result in pattern specificity at all.

We were unable to find parameters for the standard NN learning rule that allowed

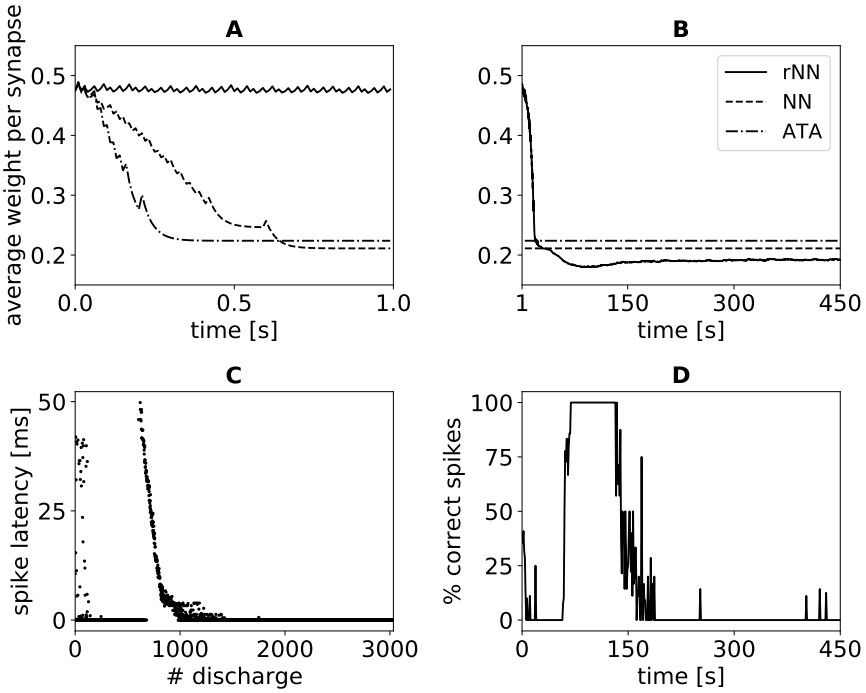

Figure 7: Effect of learning rules on synaptic weights. **A)** Both the ATA and the NN rules lead to a very rapid depression of all synapses leading to silence in the output neuron. **B)** In contrast, the average weight of all neurons declines at a much slower rate for the RNN rule. **C)** and **D)** When tweaking the ATA rule (increasing a_+ substantially) it is possible to achieve a behaviour that resembles pattern finding. For a short amount of time the output neuron becomes specific to the pattern, but loses this ability again and does not regain it.

for pattern finding, despite this rule exhibiting slower learning when compared to the ATA rule. In the case of both the ATA and NN learning rules, it seems likely that stable pattern finding relies on a precise balance of a_+ and a_-, with runaway potentiation or depression likely if the balance is wrong. By contrast, the RNN rule is automatically balanced and is not subject to this issue.

Effect of learning rule at $\Delta t = 0$

The timing of the output neuron spikes is slightly different in the original study and the replication. In the original study spike times of the input and output neurons are not restricted to fixed multiples of the timestep, so it is extremely unlikely that two neurons will spike at the same time. In Brian, the output neuron spikes at the beginning of a time step and will therefore happen in the same time bin as some input neuron spikes leading to a time difference between the spikes of $\Delta t = 0$ where the STDP rule is undefined. Brian treats these input neuron spikes as if they happened just before the output neuron spike ($\Delta t < 0$, due to the scheduling of events in Brian) and will therefore increase all those weights instead of increasing some and decreasing others. This higher number of potentiations makes it more difficult for the system to systematically depress unimportant weights in order to become selective to the pattern. If the learning rule is modified so that the synaptic weight is increased as well as depressed when input neuron and output neuron spikes happen during the same time step (effectively causing a smaller weight increase), the pattern is found earlier, at around 17 s or 850 spikes (for a time step of $10^{-4}s$) which is close to the performance for smaller time steps ($10^{-6}s$) and the original paper (both 14 s or 700 spikes). We didn't use this modified learning rule in the rest of the paper because it differs from the implementation used in the majority of modelling papers.

This difficulty to depress non-relevant input neurons is the reason for the lower success rate at a high pattern presentation frequency at large time steps (10^{-4} s) as seen in Fig. 5 **A**). The time until the pattern is found during this condition is notably longer (>30 s instead of 20 s) and points towards towards the difficulties of the system to properly depress the synaptic weights of the non-pattern neurons.

EPSP shape

The kernels used in the original study simulate a gradual increase of the postsynaptic neuron voltage as can be seen in Fig. 1. Other studies sometimes also model the effect of the presynaptic spike as an immediate jump in postsynaptic voltage instead of that gradual increase. In this system, using an immediate increase in postsynaptic voltage does not lead to stable pattern finding. This might have to to with the fact that the kernel shape and the immediate increase exhibit slightly different spike times as seen in Fig. 8.

When modelling the immediate voltage increase, one needs to set the magnitude of the voltage increase (for Fig. 8 Δu was set to 1.2). It is very difficult to find the Δu that corresponds to the same amount of voltage increase from the kernel. If the value of Δu is too small, then the output neuron stops firing after a short period of time without having gained specificity for the pattern. If the value for Δu is too high all input neurons are potentiated and the output firing rate rockets. For the scope of this paper, no value for Δu was found to induce a stable pattern finding behaviour.

Conclusion

We could successfully replicate the results from Masquelier, Guyonneau, and Thorpe [5] - a neuron with STDP synapses could reliably find a repeating spike pattern and afterwards spike only when the pattern is presented. The time when the pattern is found depends on the time step size chosen for the simulation, whereas the success

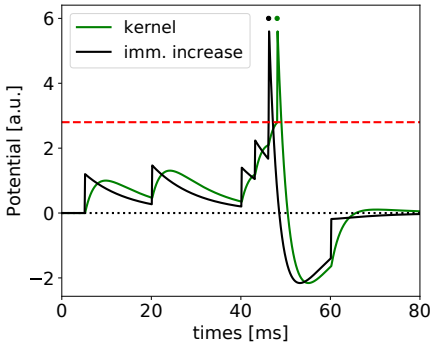

Figure 8: Choice of EPSP. For the same presynaptic spikes, the postsynaptic voltage behaves similarly, but the postsynaptic spike time is slightly different (green and black dots).

of reliably finding the pattern requires the precise learning rule and the shape of the EPSPs. Reproducing the paper was made easy by the fact that simulation parameters were stated in the text and the source code was shared by the authors of the paper on request.

Introducing discrete time steps. To run the simulations in Brian, we introduced discrete time steps. This did not affect pattern finding abilities (as long as the time steps are 10^{-4} s or smaller), but it did increase the time until a pattern was found for large time steps (10^{-4} s), but not for small time steps (10^{-6} s).

We verified that the delay in pattern finding did not stem from either the deletion of some input spikes (to avoid input neurons spiking twice in one time step) or the spike timing being at the start of each 0.1 ms time bin. When we fed those modified input spikes to the original implementation, the timing of finding the pattern was not affected. The delay therefore stems from forcing the output neuron to spike at the beginning of a time step and the associated consequences for STDP learning.

Running the simulations using discrete time steps does not negatively impact pattern finding abilities, but delays pattern finding for large time steps due to a larger number of potentiations.

Choice of learning rule. The learning rule used is one specific version of a NN STDP rule, which was not clearly stated in the original article. The comparison with other learning rules shows that the use of this particular learning rule enables the pattern finding behaviour.

The usage of the RNN rule has two interesting consequences. Firstly, it slows down the rate of synaptic weight change, since it considers fewer spike-spike interactions than other learning rules. This slows down overall weight changes significantly and gives the system more time to learn the spike sequences of the pattern. Neurons that spike together during the pattern will experience similar weight changes more often than non-pattern neurons. Over the course of hundreds of output neuron spikes - until pattern specificity arises - these synchronised weight changes lead to higher synaptic weights for the pattern neurons than for the non-pattern neurons.

Secondly, for this particular input the restrictions of the RNN STDP rule lead to the weights alternating between increase and decrease. The stabilising effect of this is most clear during the converged phase. The output neuron only spikes when a pattern is presented more or less at the same latency. The input neurons (which are forced to spike at least once every 50 ms) reliably spike at least once before or after or both. In the latter case the input neurons will always experience the same increase or decrease in synaptic weight per output neuron spike disregarding small changes due to jitter and noise. If the input neuron only spikes once per pattern, then either the increase or decrease of weight will be nearly constant whereas the respective decrease or increase

in weight will be random. On average the weights of all input neurons will slowly increase to 1 or decrease to 0 over time, resulting in a very stable system.

Choice of EPSP shape. It seems to be essential to use a kernel EPSP shape since using immediate voltage increases as the effect of an input neuron spike does not lead to learning of the pattern. This might be due to the difficulty in finding the correct parameters that create an equivalent voltage increase. It seems likely that the EPSP shape is responsible for the decrease in success, since small changes in output neuron spike times also occur when using different time step sizes and do not affect pattern finding abilities under standard conditions.

References

[1] Dean V. Buonomano and Wolfgang Maass. "State-dependent computations: spatiotemporal processing in cortical networks". In: *Nature Reviews Neuroscience* 10.2 (Feb. 2009), pp. 113–125. ISSN: 1471-003X. DOI: 10.1038/nrn2558. URL: http://www.nature.com/articles/nrn2558.

[2] Jean-Marc Fellous et al. "Discovering spike patterns in neuronal responses." In: *The Journal of Neuroscience* 24.12 (Mar. 2004), pp. 2989–3001. ISSN: 1529-2401. DOI: 10.1523/JNEUROSCI.4649-03.2004. URL: http://www.ncbi.nlm.nih.gov/pubmed/15044538%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2928855.

[3] Wulfram Gerstner and Werner M. Kistler. *Spikind neuron models*. Cambridge University Press, 2002.

[4] Dan F M. Goodman and Romain Brette. *The brian simulator*. Sept. 2009. DOI: 10.3389/neuro.01.026.2009. URL: http://journal.frontiersin.org/article/10.3389/neuro.01.026.2009/abstract.

[5] Timothée Masquelier, Rudy Guyonneau, and Simon J. Thorpe. "Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains". In: *PLoS ONE* 3.1 (2008). ISSN: 19326203. DOI: 10.1371/journal.pone.0001377.

[6] Abigail Morrison, Markus Diesmann, and Wulfram Gerstner. "Phenomenological models of synaptic plasticity based on spike timing". In: *Biological Cybernetics* 98.6 (2008), pp. 459–478. ISSN: 03401200. DOI: 10.1007/s00422-008-0233-1.

[7] Yifat Prut et al. "Spatiotemporal structure of cortical activity: properties and behavioral relevance." In: *Journal of neurophysiology* 79.6 (1998), pp. 2857–2874. ISSN: 0022-3077. DOI: 10.1126/science.1529342. URL: http://www.snl.salk.edu/%7B~%7Dzador/PDF/JNP2857.pdf.

[8] Marcel Stimberg et al. "Equation-oriented specification of neural models for simulations". In: *Frontiers in Neuroinformatics* 8 (2014), p. 6. ISSN: 1662-5196. DOI: 10.3389/fninf.2014.00006. URL: http://journal.frontiersin.org/article/10.3389/fninf.2014.00006/abstract%20http://www.ncbi.nlm.nih.gov/pubmed/24550820%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3912318.