The right delay

Detecting spike patterns with STDP and axonal delays

Arvind Datadien
ArvindDatadien@student.ru.nl, ADatadien@gmail.com
s0417491

Bachelor thesis

Supervisors:
Pim Haselager
Ida Sprinkhuizen-Kuyper

April 2010

Abstract

Axonal conduction delays are often ignored in simulations of spiking neural networks. Here, models of spiking neural networks with spike-timing-dependent plasticity, and conduction delays, are used in three simulation experiments. The first two are based on studies by Masquelier et al. [10, 11], and the third is an expansion on their work. First, it is confirmed that a neuron can learn to detect the start of a repeating spatio-temporal spike pattern. Then, we confirm that multiple neurons can learn to detect multiple patterns, when they compete with each other through lateral inhibition. Finally, we show in a new experiment that by using axonal conduction delays, we can make a neuron sensitive to specific spatio-temporal spike patterns.

1 Introduction

The brain is a network of neurons. Exciting indeed, such an organ full of potential. Now that your interest has been spiked, let's not delay any longer. After all, timing is everything.

Our sensory organs convert stimuli from the outside world into electrical signals (spikes) that can be processed by neurons in the brain. When a stimulus is presented, it will result in multiple neurons that fire at certain times. A group of neurons that fire at certain times, create what we call a spatio-temporal spike pattern, which may be transmitted to other neurons. Presenting a stimulus repeatedly should result in the same spike patterns arriving at the same neurons each time. If a neuron could learn to detect a spike pattern, it could (among other things) trigger an appropriate response to the stimulus. Fortunately for us, it can.

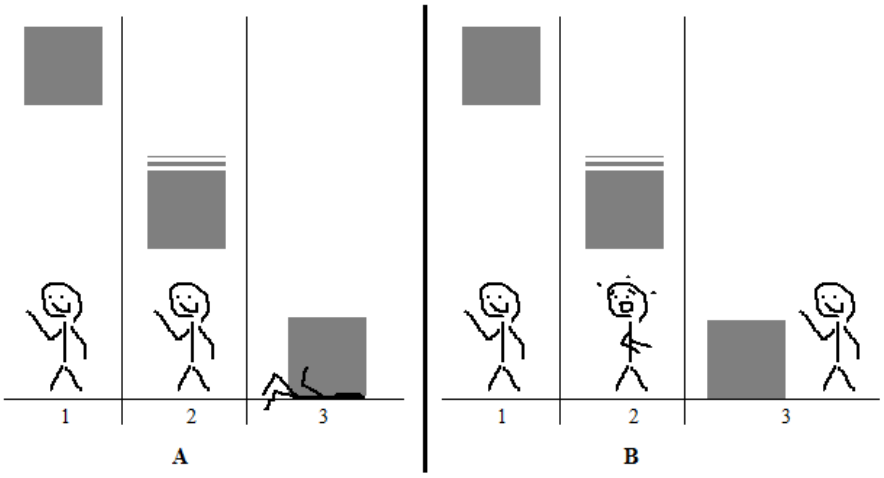

Figure 1: A) A stick figure is crushed by a falling object. B) A stick figure avoids a falling object. The movement of an object in space, over time, may be converted by the retina into a spatio-temporal spike pattern. As shown here, it is sometimes convenient to detect such patterns early. STDP allows neurons to learn to detect the start of a spatio-temporal spike pattern, so that fatal outcomes such as the one in part A of this figure may be avoided.

It was shown in [10] that a neuron can learn to detect a spatio-temporal spike pattern, by using a learning mechanism called Spike-Timing-Dependent Plasticity (STDP). This neuron will learn to fire only during the spike pattern it has learned to respond to, and not when it receives randomly timed spikes. Also, the neuron will learn to fire earlier and earlier into the pattern, until it fires near the very start of it.

We can imagine many practical applications for the detection of spike patterns in the brain. One theoretical example is the visual detection of movement of an object. If an object moves from top to bottom, it may first activate cells at the top of the retina, then the middle, and then the bottom (or the reverse if you take into account the lens). This may cause retinal cells to create a spatio-temporal spike pattern that corresponds to this downward motion, and would be transmitted to neurons in the visual cortex. After repeatedly seeing objects moving from top to bottom, some cells in the visual cortex may learn to detect the corresponding spike pattern, and thus learn to fire during downward motion. As we can see in Figure 1, having neurons that can detect this may save one's life.

Now, it is clear that finding the start of a spatio-temporal spike pattern can be very useful, but what if we want to detect not just the start, but the entire pattern? After all, two patterns may have similar beginnings, but very different endings. It was shown in [11] that this can be achieved by using multiple neurons that inhibit each other. Such competition between neurons prevents them from all firing simultaneously near the start of a pattern. Instead, one neuron will fire near the start, one several milliseconds after, and another some milliseconds after that. The sequential firings of these three neurons will then signify the occurrence of the entire pattern. If multiple patterns are presented,

the inhibition between neurons also allows certain neurons to detect parts of one pattern, and other neurons parts of another pattern. In this way, given enough neurons, all patterns will be detected.

In this study, I have conducted three simulation experiments. The first experiment is an attempt to detect spatio-temporal patterns in a Spiking Neural Network (SNN) using STDP, and is based on the work of [10]. The second experiment is inspired by [11], and attempts to detect multiple patterns using a SNN, STDP, and lateral inhibition between the output neurons. Software was created to conduct these experiments with a different neuron model than that used in the original studies, a different STDP rule, and while taking into account axonal conduction delays.

The third experiment is entirely new. Here, the effect of conduction delays (the difference between the moment a neuron fires, and the moment that the corresponding spike arrives at another neuron) is investigated. I expect that, using these delays, we can create a neuron that is sensitive to patterns of a certain type. Then, when presenting two neurons with two different patterns, we should be able to determine which neuron will detect which pattern.

To summarize, the tasks to be learned in the three simulations conducted in this study are the following:

1. Detect the start of a spatio-temporal spike pattern.
2. Detect multiple spike patterns.
3. Detect only a specific spike pattern.

If these tasks can be performed by an artificial neural network that shares many characteristics with biological neural networks, then it is likely that the tasks can be performed in the same way in certain areas of the brain. In this way, simulation studies allow us to test or create hypotheses about the workings of the brain.

In this paper, I will first discuss the basics of spiking neural networks and spike-timing-dependent plasticity. Then, the software and methods that were used to conduct the experiments will be described. After that, the three experiments and their results will be discussed. Finally, I will talk about some interesting findings concerning STDP, and possible future work.

1.1 Spiking neural networks

According to Maass [9], three generations of artificial neural networks can be distinguished if we look at the computational units that are used in them. The first generation consists of McCulloch-Pitts neurons, also called perceptrons, and give only digital output. The second generation use neurons with activation functions to provide analog output. These values may be interpreted as the equivalent of the mean firing rate of a biological neuron. It has however been shown that the brain does not solely rely on firing rates to work, but also on individual spike timings. The third generation of neural networks are called Spiking Neural Networks (SNN), and do model these individual spikes.

A SNN contains neurons with a membrane potential (an activation level) and axons that connect to the dendrites of other neurons through synapses. Once a neuron's membrane potential has been increased to a certain threshold,

it is said to "fire", and an action potential (a spike) travels from the cell body, along the axon, to the synapse. Neurotransmitters are then released at the synapse, causing a change in the postsynaptic neuron's membrane potential. An excitatory synapse will increase the postsynaptic potential, possibly causing the neuron to fire, while an inhibitory synapse will do the opposite. The strength (or weight) of a synapse may vary, meaning that a spike may cause much or little change in a postsynaptic neuron's membrane potential.

The delay that is caused by a spike traveling along an axon is often disregarded in models of spiking neural networks. However, Izhikevich [7] showed that the presence of axonal conduction delays (also known as transmission delays) of various lengths might be very important to the functioning of the brain. Conduction delays between neurons, can be as small as 0.1 ms, or as large as 44 ms. Because the brain can reproduce spike timings with sub-millisecond precision, it seems likely that these delays should not be ignored in simulations.

1.2 Spike-timing-dependent plasticity

One of the great strengths of the brain is its ability to learn. Although there are effective learning mechanisms for the first and second generation of neural networks, much research is still done on finding mechanisms with which a spiking neural network may learn. Learning may occur as a result of the change of synaptic weights, but also the creation of new, or the pruning of old connections, the addition, removal, or alterations of neurons, the adjustment of axonal conduction delays, or possibly other changes. Some may be biologically plausible, meaning that they are observed in the brain, while others may not.

One learning method that has been observed to occur in the brain is Spike-Timing-Dependent Plasticity (STDP). It is the alteration of a synaptic weight, based on the difference between firing times of two connected neurons, or when we take into account conduction delays, the difference between the arrival time of a spike at a neuron, and the firing time of that neuron. Weight changes caused by STDP remain over longer periods of time. A decrease in weight is therefore called Long Term Depression (LTD), and an increase is called Long Term Potentiation (LTP). Mechanisms for short term synaptic plasticity also exist, but will not be used or discussed in this paper.

In 1949, Hebb [4] stated that "When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased". This effect has been observed to occur in neurons as a result of STDP. STDP is therefore sometimes called a Hebbian learning rule [2].

Multiple STDP rules have been observed [2]. The one that will be discussed here is often found in excitatory to excitatory connections, and shown in Figure 2. When neuron A fires, a spike travels along the axon, to the synapse connected to neuron B. When it arrives there at time t_1, it may cause neuron B to fire shortly after, at time t_2. If this happens, the interval $t = t_2 - t_1$ is positive. When looking at the STDP rule in Figure 1, we see positive values for $t > 0$, so the weight of the connection from A to B should be increased. The smaller the interval, the more likely it was that A was the cause of B's firing, so the larger the weight gain, or LTP.

Suppose neuron A fires. A spike travels along the axon towards neuron B,

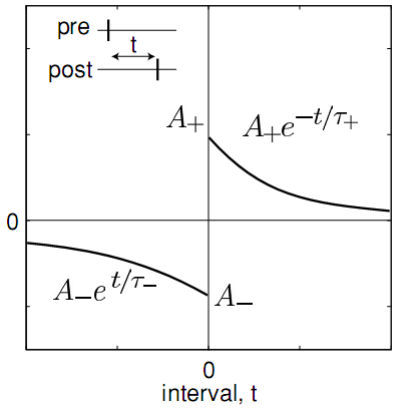

Figure 2: A classic STDP rule, copied from [7]. The y-axis shows the change in synaptic weight Δw. The x-axis shows the interval t in milliseconds, which is the last postsynaptic firing time minus the last presynaptic spike arrival time of a connection between two neurons. A positive value for t causes LTP according to A_+e^{-t/τ_+}, and a negative value causes LTD according to A_-e^{t/τ_-}. The smaller the interval t, the larger the resulting weight change. The variables A_+, A_-, τ_+, and τ_- are manually chosen parameters.

but before it arrives there at time t_1, B already fires at t_2. This can happen because B is connected to other neurons that can make it fire. So the spike from A was too late to make B fire, the interval $t = t_2 - t_1$ is negative, and now according to the STDP learning rule, the weight from A to B will be decreased. LTD will occur.

An important requirement for the STDP rule is that the surface beneath the graph for the LTD part ($t < 0$) should be greater than the LTP part ($t > 0$). This allows a neuron to disregard random inputs. Suppose neuron A's spikes arrive at random times t_1 at neuron B, which fires at times t_2. Because times t_1 are random, on average, LTD ($t_1 > t_2$) will occur as often as LTP ($t_1 < t_2$). If the LTD part of the STDP rule is greater than the LTP part, the weight from A to B will eventually be reduced to zero. It should be noted that in this example, neuron A does not necessarily fire at random times. Rather, neuron A's firing times are uncorrelated to those of neuron B, and therefore, from the perspective of neuron B, seem random. There may very well be a neuron C whose firings times are correlated to those of neuron A, so that from the perspective of C, A does not fire randomly.

In its normal unsupervised form, STDP can result in classical conditioning, or Pavlovian or respondent conditioning. When STDP is modulated through feedback, it can be a mechanism for operant conditioning, also known as instrumental conditioning. This is discussed in an article by Izhikevich [8], and another by Florian [3]. In this paper however, we will not use modulated STDP. We will use standard STDP, in a way that is discussed in detail in the following section.

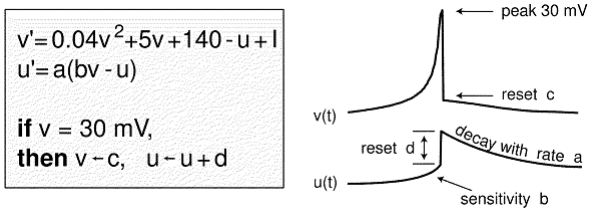

Figure 3: Izhikevich Simple Model, copied from [5]. A model of a neuron's membrane potential over time. The variable v is the membrane potential, and u is a recovery value. $v' = dv/dt$ and $u' = du/dt$. The variables a, b, and c are manually chosen parameters; typical values for them are discussed in [5]. The variable I is the incoming potential, usually from spikes received from other neurons. All values are chosen so that time is in ms scale, and weight is in mV scale. The neuron is considered to have fired when its membrane potential reaches 30 mV, after which the values of v and u are reset.

2 Materials and methods

To conduct the three experiments, I have constructed a Java program that simulates a spiking neural network with STDP. It allows easy inspection of the firing times and weight changes, as they occur during simulation.

Neurons in the network are simulated with Izhikevich' Simple Model [5] which is shown in Figure 3. This model can exhibit many characteristics of biological neurons, at a reasonably low computational cost. By simply changing parameters, different types of neurons can be simulated. Two alternative models are the more simple Leaky Integrate-and-fire model which was also used in the papers by Masquelier et al. [10, 11], or the more complex Hodgkin-Huxley model. For a more elaborate comparison of models of spiking neurons, see [6].

In the following experiments, unless stated otherwise, the output neurons were configured to behave as regular spiking neurons, by using the parameter values $(a = 0.02)$, $(b = 0.2)$, $(c = -65)$, $(d = 6)$. For the input neurons, the values $(a = 0)$, $(b = 0)$, $(c = -65)$, $(d = 0)$ were used to obtain a neuron type that will fire instantly when it receives a large current I of 100 mV for 1 ms as input, and do nothing else. The regular spiking neuron model was not suitable to use for the input neurons, because it could not always fire exactly when needed. It was too realistic for our purposes.

Connections between neurons are determined at the start of the simulation, they can not be added or removed when the program is running. The connections can be excitatory, meaning that spikes there will increase the value of I, or inhibitory, meaning the opposite. The axonal conduction delay for each connection remains the same during simulation, but weights of connections may change. Any neuron in the network can have a connection to any other neuron, or to itself. Every such connection has a minimum conduction delay of 1 ms, meaning that a spike will arrive at the target neuron 1 ms after the source neuron has fired. Circular connections (from A to B, and from B to A) are possible. Multiple connections from A to B are not used.

The network algorithm used here is synchronous, or "clock-driven", meaning that there is a clock that is advanced by a small time step every tick, after which

each neuron's state is updated to reflect possible changes. The alternative is an asynchronous, or "event-driven" algorithm [1], in which updates are only made when an event (such as the firing of a neuron) occurs. Such a system can have precise firing times that are not limited to the size of a chosen time step, and may run faster at times when not many events occur. However, they are more difficult to implement, especially when using transmission delays. I did not use this type of algorithm because the increased timing precision seemed initially unnecessary, although, as will be seen in the second experiment, higher precision would have been useful.

Each tick of the simulated clock represents 1 ms of real time. After each tick, the new state of the network must be determined. This is done in the following steps:

- For each input neuron that we want to fire at this time, add an incoming potential of 100 mV.
- For each neuron, advance all the spikes that are currently traveling along each of its axons by 1 ms. If a spike arrives at a neuron, add an incoming potential to that neuron equal to the weight of the connection. If a spike arrives at a neuron, apply STDP. The weight from the neuron that fired will (usually, depending on the STDP rule) be decreased (LTD will occur).
- For each neuron, update its membrane potential, and note if it has fired. The new membrane potential is determined by using Euler's method to approximate the solutions to the differential equations given in Figure 3. In a MATLAB program by Izhikevich (available on his website), two steps of 0.5 ms were used. For extra precision, I used 5 steps of 0.2 ms.
- For each neuron that has fired during this time step, create a spike on each of its axons, and apply STDP. The weights from neurons having connections to the firing neuron will be increased (LTP will occur).
- For each neuron, apply the net change to its weights, according to the determined LTP and LTD values.

We see that the final weight change that results from STDP is determined in two separate steps:

1. When a spike from neuron A arrives at neuron B, we look at the last time B has fired. This will be a point in the past, so the spike will always be considered to be "too late", and so this step will result in Long Term Depression (LTD).
2. When neuron B fires, we look at the last time a spike from neuron A arrived at B. Because the firing time of B will always be greater than the last time a spike arrived, this step will result in Long Term Potentiation (LTP).

During testing, at first the STDP rule in Figure 2 was used. However, determining suitable values for the parameters proved to be a problem. Masquelier et al. used the values $\tau^+ = 16.8$, $\tau^- = 33.7$, $A_+ = 0.03125$, $A_- = 0.85 * A_+$ with a maximum post-pre time window of $[-7\tau^-, 7\tau^+]$. These values did not

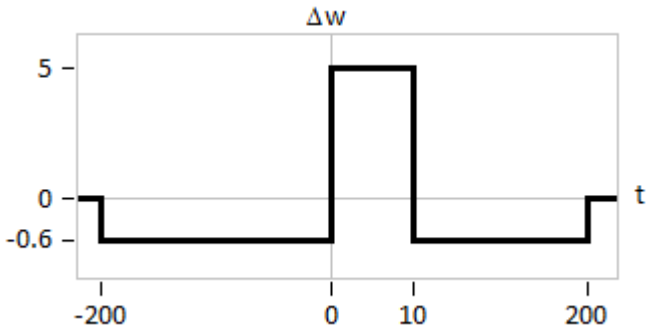

Figure 4: The STDP rule that was used in the experiments here. It is based on the STDP rule that was used in [12]. The variable t is the postsynaptic firing time minus the presynaptic spike arrival time, and Δw is the resulting change in weight. The values $\Delta w = 0.01 * 5$ for $(0 < t < 10)$, and $\Delta w = 0.01 * -0.6$ for $(-200 < t <= 0)$, and $(10 <= t < 200)$ were used, based on good results with empirical testing. The numbers that are mentioned don't match the graph exactly.

work for the network configuration used in this study. Hopes that the values used by Izhikevich (2006) ($\tau^+ = 20$, $\tau^- = 20$, $A_+ = 0.1$, $A_- = 0.12$) would perform better, were crushed when each test resulted in an inactive output neuron due to low synaptic weights. Experimenting resulted in some parameters that worked reasonably well, but eventually the best performance was achieved with an STDP rule with a different form, as can be seen in Figure 4. It was inspired by an article by Nessler, Pfeiffer, and Maass [12].

The STDP rule in Figure 2 has basically the same attributes as the one in Figure 4; it has LTD for negative values, and LTP for positive values of t. One difference is that when t is positive and becomes too large, the rule again states that LTD should occur. The reason for this is that, when a spike arrives at a neuron and after a longer period of time, that neuron fires, the effect of the spike will have disappeared and therefore it could not have caused the firing. Tests with the neuron configuration used here, showed that a presynaptic spike causes an increased membrane potential in the postsynaptic neuron for about 8 to 10 ms. After that, the membrane potential has returned to its resting state. The size of the synaptic weight does not affect this period. Therefore, we can assume that if a presynaptic spike arrived more than 10 ms before the postsynaptic neuron fired, that spike did not (help) cause the neuron to fire, and the change in weight should not be positive but negative.

At the end of each time step, the net weight change for each neuron based on the STDP rule, is applied. The weights are then clipped to stay within a maximum and minimum value. For excitatory connections, a minimum weight of 0 mV was chosen, and a maximum of 5 mV, although in the last experiment, a lower maximum value of 2 mV was used. A high maximum value for excitatory connections resulted in neurons firing often more than once during pattern presentation, but also resulted in more false positive firings. A maximum value that was too low resulted in many cases where the pattern could not

be learned. For inhibitory connections, STDP was not applied, so the weights were not changed and therefore no clipping was needed.

Now that the software has been described, we will discuss the simulations that were conducted with it.

3 Experiments

In this section, three simulation experiments are described. In the first experiment, we allow a SNN to learn to detect spatio-temporal spike patterns. The following two expand on this; multiple patterns are detected in the second experiment, and specific patterns in the third.

3.1 Detecting the start of a spatio-temporal pattern

In this first experiment, the goal was to let a SNN with STDP learn to fire only when a spatio-temporal pattern was presented, as was done by Masquelier et al. [10]. When no pattern was presented, the network was given random input, and should not fire.

The network consisted of 100 input neurons, and 1 output neuron. Each of the input neurons was connected to the output neuron with an initial randomly determined weight between 3 and 5 mV, and a time delay of 1 ms.

The network was given cycles of input similar to those in Figure 5 which lasted for 100 ms. Each cycle consisted of 5 parts of 20 ms: a random part, the pattern, and three more random parts. Each part contained an average firing rate of 20 Hz for each of the 100 input neurons. This means that each neuron has a probability of $0.02 * 20 = 0.4$ to fire during each part, so that on average, each part of 20 ms contained $0.4 * 100 = 40$ firings. The only difference between the pattern and the random parts was that after each cycle, the random parts were recreated with new random firing times, while the pattern remained exactly the same. The cycles were presented continuously with no breaks in between, until the simulation was stopped.

Twenty trials were conducted. The results can be seen in Table 1. The output neuron learned to fire during the pattern, and remain quite silent during random input, in all 20 trials. This does not mean an absolute 100% success rate; failed attempts to learn the pattern were observed during testing, but by chance no failures occured during these 20 trials. In the 5th and 13th trial, the large number of firings during the second random input part, occured right at the beginning of that part. They were obviously caused by the pattern that preceded it. We see that in some trials, the neuron learned to fire twice during a pattern. Interestingly, in these cases the number of false positive firings also increased. It is not clear to me why most of the false positive firings occured in the random input period that preceded the pattern.

Learning was sometimes finished after 100 cycles, or pattern presentations, while at other times the weights did not fully settle until 1400 cycles. A learning period of 400 to 500 cycles seemed average.

It should be noted that these results were obtained after much tweaking of parameters, such as the STDP function, the maximum weight for neurons, and the firing rate for input patterns.

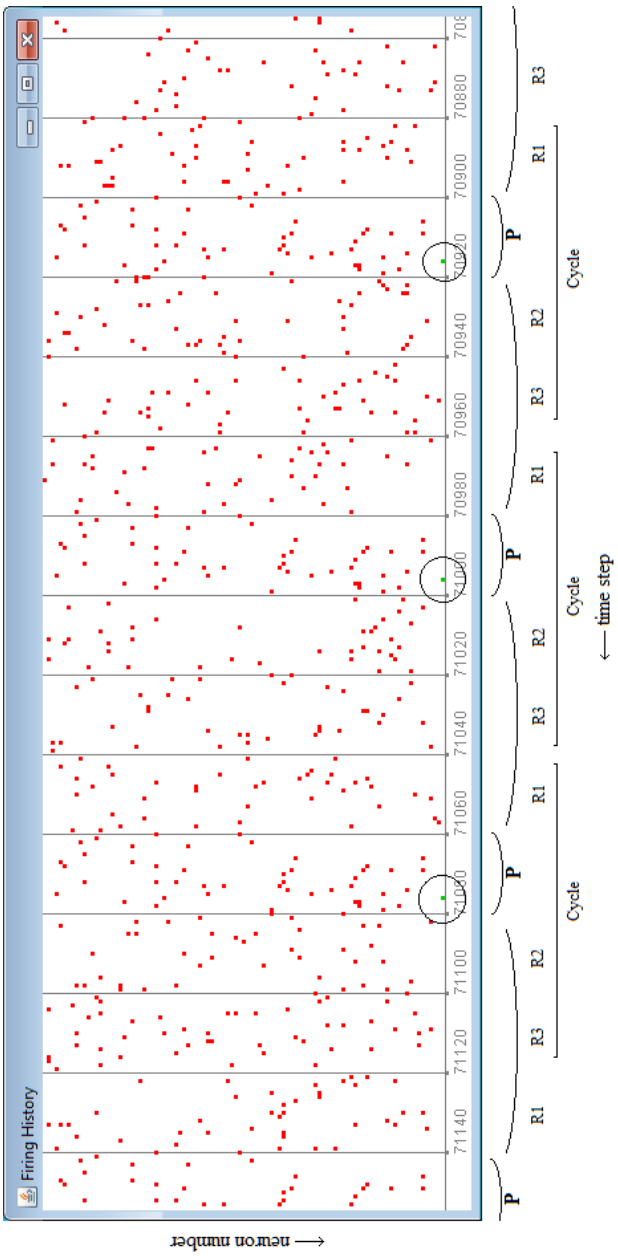

Figure 5: Firing times. The y-axis shows the neuron, the x-axis shows the time step (left is new, right is old). Red dots are firings by excitatory neurons, green dots are firings by inhibitory neurons. In this case, the top 100 neurons are excitatory input neurons. The bottom neuron is an inhibitory output neuron. Each cycle consists of 20 ms random input R1, 20 ms of the pattern P, 20 ms random input R2, and 20 ms random input R3. In this case the order of these parts remains the same, they are not shuffled. We see that the output neuron fires during every occurrence of the pattern P.

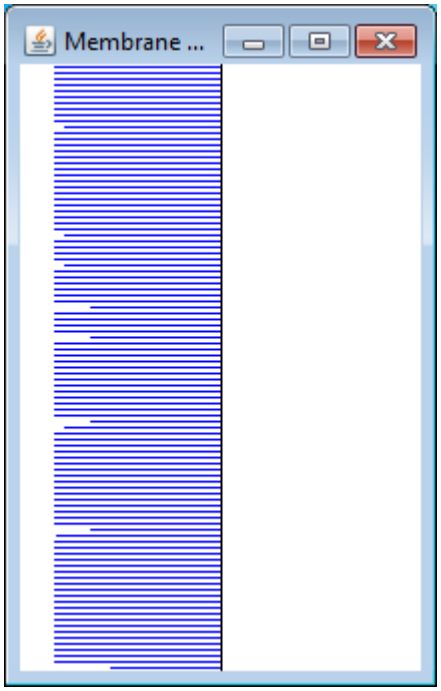

Figure 6: Membrane potentials at a point in time. Each line represents the current membrane potential of a neuron. The far left corresponds to -100 mV, the far right to 100 mV. The top 100 neurons are input neurons. The bottom one is an output neuron. Here, most neurons are at their resting potential at about -70 mV. No neurons are currently firing (they are all < 30 mV), but the output neuron is probably about to fire.

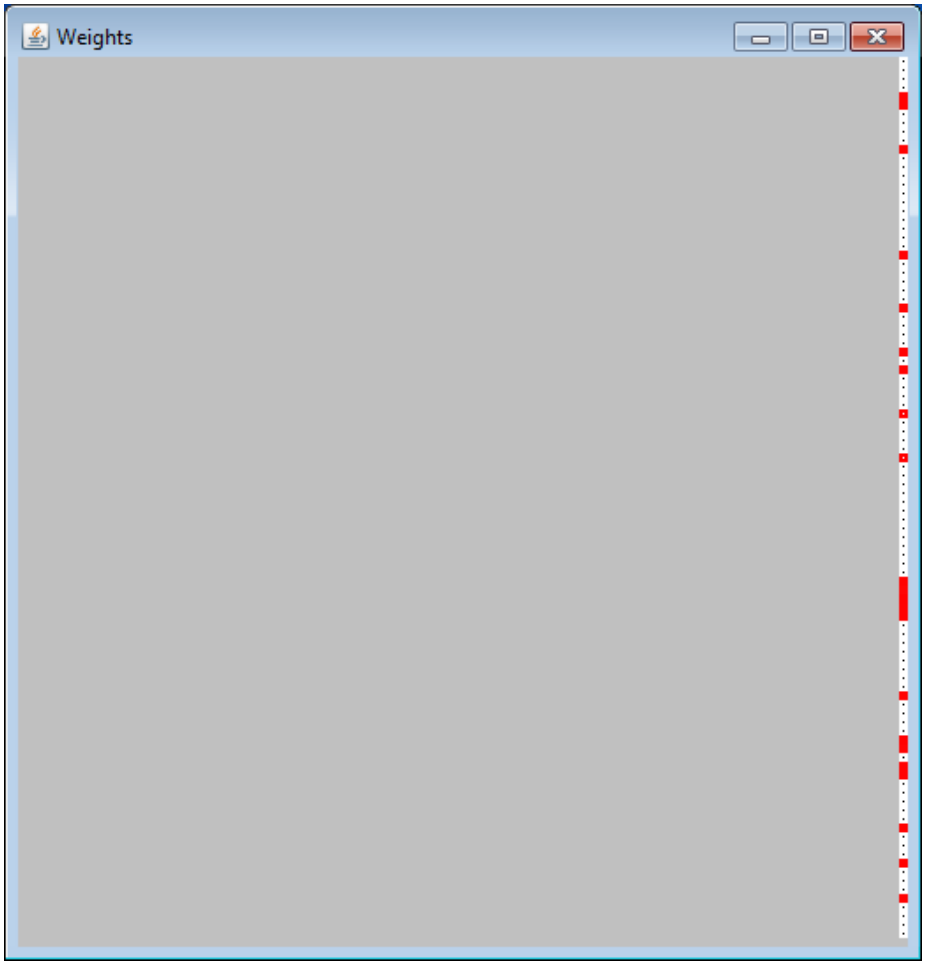

Figure 7: Synaptic weights at a point in time. Values on the X and Y axes correspond to neurons. Here there are connections from neuron 1-100, to neuron 101. A gray square means there is no connection between the neurons, a white square means that there is a connection, but its weight is at 0%, a red square indicates an excitatory, and a green square an inhibitory connection. Various degrees of brightness of red or green indicate weights between 0% and 100%, but all connections here are at 100%, because the network has stabilized. Inhibitory connections are not present in this figure.

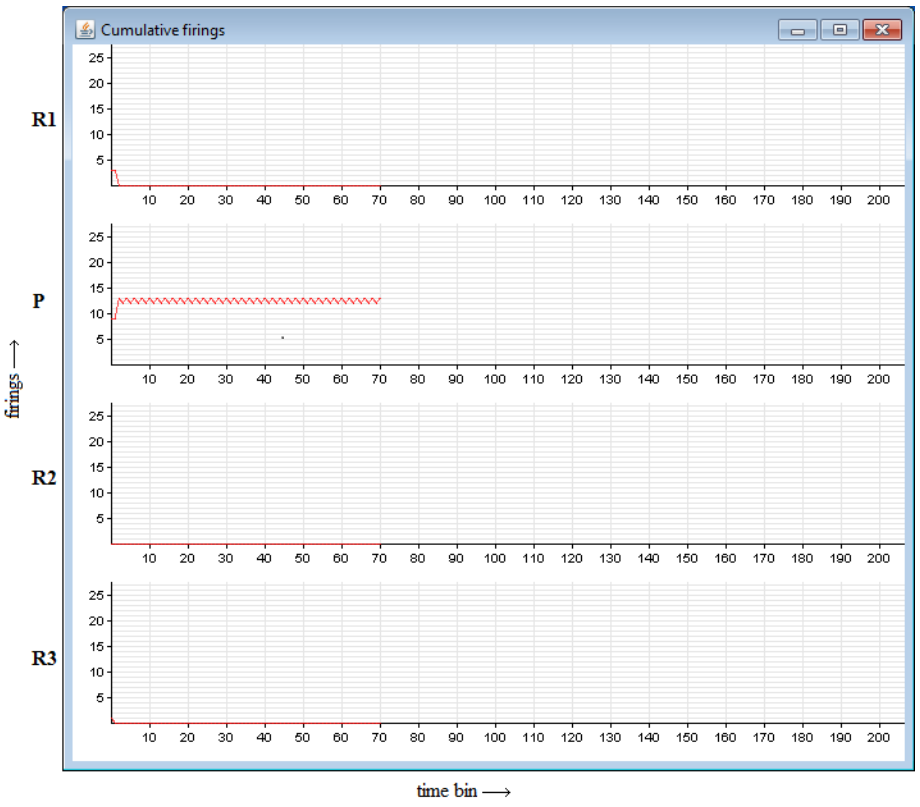

Figure 8: Firings of output neurons per time-bin. The four cycle-parts of Figure 5 are represented by these four graphs. Each graph shows the number of times an output neuron fired during the corresponding cycle-part, per time-bin of 1000 ms. Here, only one output neuron was used, so there is only one line per graph. Pattern P occurred 12 times during half the time-bins, and 13 times during the others, because the 80 ms cycles fit 12.5 times in the 1000 ms time-bins. We see that the output neuron never fired during the periods with random input, and always during the pattern P. The simulation was stopped after 70000 simulated ms.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
Random 1	0	40	4	6	84	1	0	0	13	23
Pattern	1000	1957	995	1000	1206	999	1002	1000	1985	1971
Random 2	0	0	0	0	698	0	0	0	0	0
Random 3	0	0	1	0	3	0	0	0	0	0
Random 4	0	3	0	0	9	0	0	0	2	1

	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20
Random 1	3	43	36	6	59	49	6	5	2	4
Pattern	997	1954	959	994	1931	1948	994	995	998	995
Random 2	0	0	1000	0	0	0	0	0	0	0
Random 3	0	1	1	1	3	0	0	0	0	0
Random 4	0	7	4	3	16	3	0	1	1	3

Table 1: The result of twenty trials where a pattern surrounded by 4 periods of random input was presented to an output neuron. The numbers correspond to the number of times the output neuron fired during each part of the cycle, counted over 1000 cycles. These numbers were recorded after the 2000th cycle, when learning was completed, and the weights were stable.

In conclusion, we can clearly see that the output neuron learned to fire during the pattern, and remain silent in most other cases. Although not visible in Table 1, it was also observed that the output neuron learned to fire increasingly early into the pattern, up until about 5 to 10 ms after the start of the pattern.

3.2 Detecting multiple spatio-temporal patterns

In this second experiment, based on a different paper by Masquelier et al. [11], we will see if lateral inhibition between multiple output neurons will allow them to fire during different (parts of) patterns. This experiment will expand on the first one in three steps. First we will see what happens when we simply increase the number of output neurons. Then, we will add inhibitory connections between these output neurons. Finally, we will increase the number of patterns per cycle, and see what happens.

At first, I took the configuration of the first experiment, but with three instead of one output neuron. Each output neuron received an excitatory synaptic connection from each of the input neurons. At this point, there were no connections between the output neurons. The network was again repeatedly given an input cycle which was 100 ms long, and consisted of 5 parts of 20 ms: two random parts, the pattern, and two more random parts.

Twenty trials were conducted. Because each trial contained 3 output neurons, after twenty trials there were a total of $20 * 3 = 60$ output neurons. In the end, 59 out of 60 output neurons successfully learned to fire during the pattern, and almost never during random input. The results were (not surprisingly) similar to those of the previous experiment. In most cases the output neurons fired synchronously near the start of the pattern, and all three had nearly identical weights from the input neurons.

Following this, I added inhibitory connections between the three output neurons. These weights were effectively set at -25 mV, so that each output neuron should be able to prevent its neighbors from firing in most cases. The connections had a 1 ms delay, meaning that the spike of a firing output neuron would arrive at the other output neurons one millisecond later. STDP was

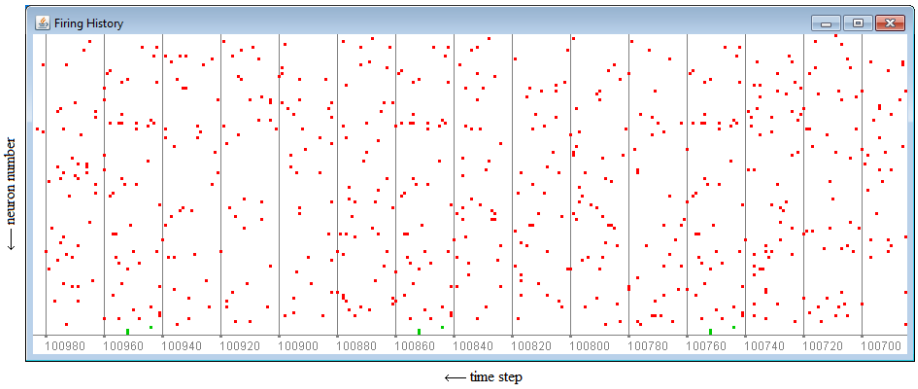

Figure 9: Firing times with 1 pattern and 3 competitive output neurons at the bottom. We see that the top output neuron (third from the bottom) fires first during pattern presentation. The other two output neurons fire about 8 ms later.

not applied to these inhibitory connections. As before, only one pattern was presented to the network.

As can be seen in Figure 9, the neurons would often learn to fire at different times within the pattern; one neuron near the start of the pattern and another about 5 to 10 ms later. Unfortunately, when two neurons fire at (almost) the same time, the lateral inhibition loses its effect because the inhibiting spikes arrive too late. As was described in "Materials and methods", a clock-driven algorithm was used for these simulations, with a time resolution of 1 ms. Because of this, it was not uncommon for two output neurons to fire simultaneously, resulting in ineffective lateral inhibition. In a biological neural network where time is continuous, and conduction delays can be as small as 0.1 ms, this would not be such an issue.

Finally, multiple patterns were presented to the output neurons. Because lateral inhibition was sometimes ineffective, as described above, I used five instead of three output neurons. The network was repeatedly given an input cycle which was 120 ms long, and consisted of 6 parts of 20 ms: one random part, the first pattern, two random parts, the second pattern, and one more random part. After each cycle presentation, the parts were shuffled, meaning that their position within the next cycle was randomly determined.

Testing showed that many neurons would indeed learn to fire at different parts of the patterns (see Figure 10 and 11). Looking at the network output, one might know when either of the two patterns was presented at a certain time. In one case for example, the firings of neurons 2 and 4 would indicate that pattern A was present, and the firings of neurons 1, 3, and 5 would indicate the presence of pattern B.

To conclude, multiple spatio-temporal spike patterns can effectively be detected by using lateral inhibition between output neurons, but it should be noted that a relatively high time resolution, and a low axonal conduction delay between the output neurons is required for this to work effectively.

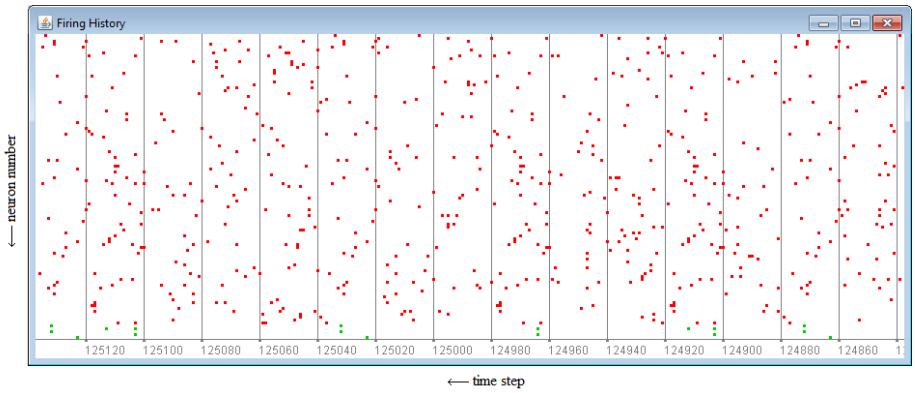

Figure 10: Firing times. Two patterns were presented to 5 output neurons. There was lateral inhibition between the output neurons. Neurons 1, 3, and 5 learned to fire during pattern B, and neurons 2 and 4 during pattern A.

3.3 Detecting specific types of spatio-temporal patterns

This final experiment is new, and an expansion on the other two experiments. Previously, all conduction delays were set to a minimum of 1 ms. Here, we will investigate if the delays can be used to make output neurons sensitive to certain types of patterns.

For this experiment, I created two spatio-temporal spike patterns (see Figure 12), and set the delays between the input neurons and the first output neuron so that they would match the first pattern. The same was done for the second output neuron and the second pattern. The two patterns were presented each cycle in random order, together with four parts of random input. All parts lasted for 20 ms, making each cycle 120 ms long.

An example may clarify the concept of delays that "match" the firing times of a pattern. When input neuron A fires at 0 ms, and input neuron B at 6 ms, the delay from neuron A to the output neuron would be 7 ms, and the delay from neuron B to the output neuron would be 1 ms, so that both spikes would arrive at the output neuron simultaneously, at 7 ms.

Results showed that in every trial, each neuron fired at 100% of the presentations of its matching pattern (an example can be seen in Figure 12). However, because the two patterns are spatially the same (they use exactly the same input neurons), the weight distributions of the two neurons were similar, and this caused the neurons to also fire during the non-matching patterns in nearly 30% of their occurrences. Although the non-matching delays made it more difficult to fire during the non-matching pattern, the highly similar weight distribution apparently overcame this obstacle. Fortunately, decreasing the maximum weight from 5 to 2 mV eliminated the problem entirely; all false positive firings disappeared. So, setting the right conduction delays succesfully made the neurons sensitive to a specific pattern. To test the robustness of this method, two more tests were conducted.

In the first test, only half of the connections to input neurons that fired during the patterns had matching delays. The other delays were set at 1 ms. Here we saw similar results as before, but with more false positive firings. The

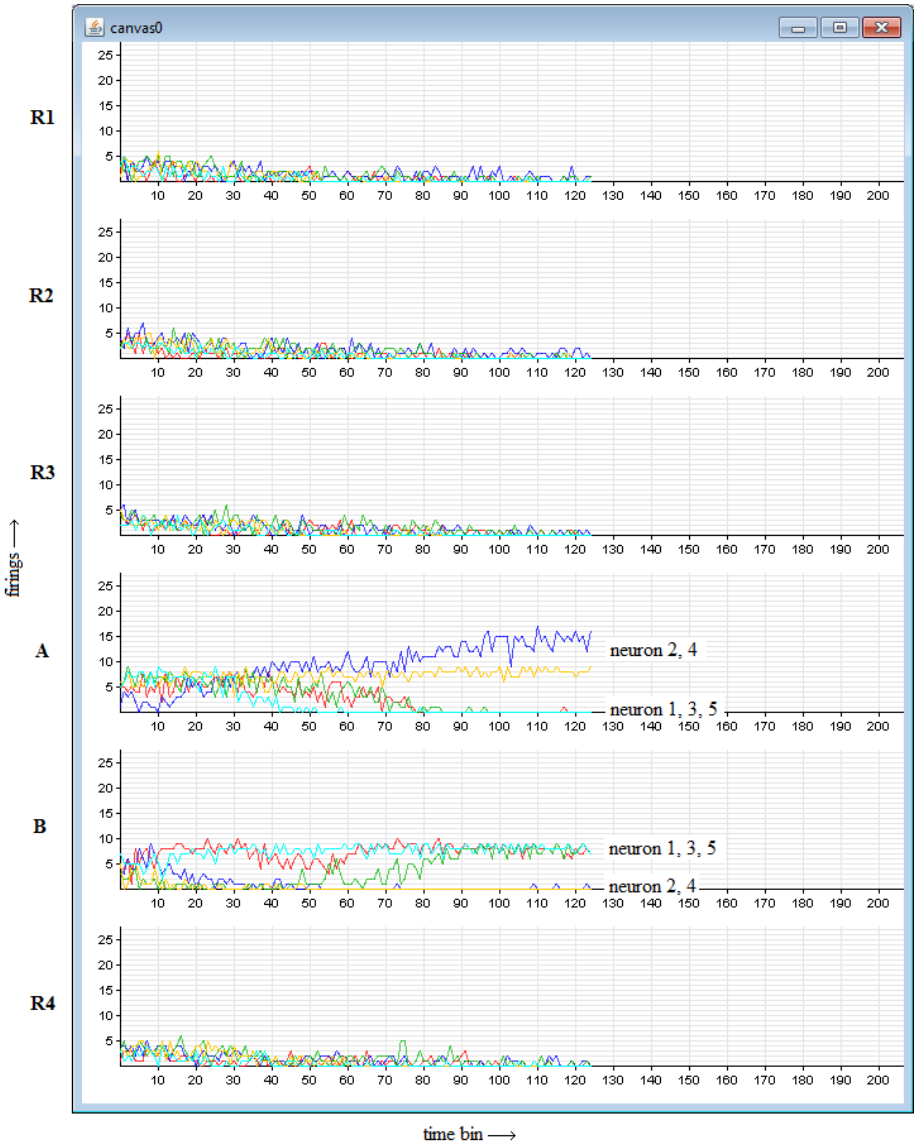

Figure 11: Firing results. Each graph shows the number firings of each output neuron per time-bin of 1000 ms, during a specific cycle-part. The top 3, and the bottom graph correspond to random input periods, the others correspond to pattern A, and B. Here there is lateral inhibition between the output neurons. We see that output neurons 2 and 4 (blue and yellow) fired during pattern A, and output neurons 1, 3, and 5 (red, green, and cyan) fire during pattern B. The simulation was stopped after about 124000 simulated ms.

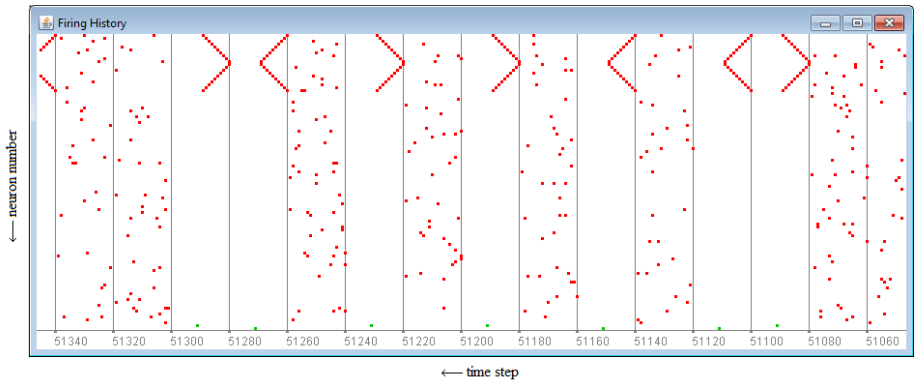

Figure 12: Firing times. The conduction delays of each output neuron match one of the patterns. We see that each output neuron fires during a different pattern.

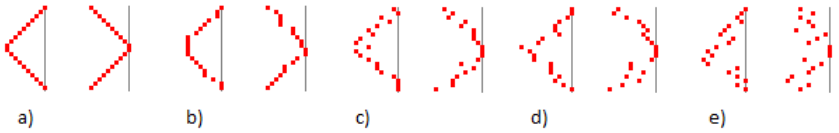

Figure 13: Input patterns A, and B, with varying levels of jitter (0, 1, 2, 3, and 4 ms)

neurons fired at 100% of the occurrences of their matching pattern, and at almost 50% of their non-matching pattern. Reducing the maximum weight from 5 to 2 mV again removed nearly all the wrong firings, however in this case, the successful firing rate dropped from 100% to about 78%. The weights were apparently not high enough to ensure a firing at every occurence of the matching pattern. Still, this shows that the delays do not need to match the entire pattern for it to be recognised by the intended neuron.

In the second test, we look at the performance when patterns become distorted. Each firing was randomly moved forward or backward in time, from 0 ms to a maximum jitter value (see Figure 13). Delays that matched the entire original patterns were used, and the maximum weight was set to 2 mV. The success rates with jitter values of 1, 2, 3, and 4 ms were tested over 6 trials each time, and can be seen in Table 2. We see that as jitter increases, the number of dead neurons (neurons that didn't learn the pattern and stopped firing) also increases. The performance of the neurons that still learned to fire during occurrences of their matching pattern, decreased, from 99% with 1 ms jitter, to 72% with 4 ms jitter.

In conclusion, we have seen that axonal delays in combination with STDP, can indeed make neurons sensitive to specific spike patterns. The delays do not need to match the pattern entirely, and the pattern does not need to match the delays precisely. In these cases, although performance drops, neurons can still often detect their matching pattern.

Jitter	Dead	Alive	Success rate	Success rate among living neurons
1 ms	0	12	99%	99%
2 ms	0	12	93%	93%
3 ms	2	10	65%	77.9%
4 ms	9	3	19%	72%

Table 2: The first two columns show the number of live and dead neurons after learning. The last two columns show the percentage of times that a neuron fired during its matching pattern. For each jitter value, six trials were conducted, and in each trial, two output neurons were used (one for each pattern). Therefore, for each jitter value, there are 12 neurons.

4 Discussion

We have seen that the results of Masquelier et al. are robust; even with an entirely different implementation, the results are largely the same. We were able to detect the start of a spatio-temporal spike pattern using STDP in a SNN. It was also possible to detect multiple patterns, and multiple parts of patterns, by using lateral inhibition.

In the third experiment, we saw that by using conduction delays and STDP, we can create a neuron that can detect specific spike patterns. It was seen that the spike patterns don't need to match the delays exactly; jittered patterns lower the performance, but can still be detected to some degree. Matching delays cause the firings of a pattern to arrive simultaneously at an output neuron, and so will usually make the output neuron fire, even if the weights are relatively low. Therefore, STDP may not always be needed to increase the weights of connections that are used in a pattern, but it is needed to lower the other weights down to zero, so that the output neuron will not fire during periods of random input.

Concerning STDP, the rule used in this paper (see Figure 4) has a limited window of $(-200 < t < 200)$ in which it can cause a change in weights. The traditional STDP rule used by Masquelier and Izhikevich (see Figure 2), also loses its effect (but gradually) as t becomes very large or small. At first, this limitation seemed unnecessary to me, but it actually is. It is required, in order for a neuron to remember its weights for a pattern, even when the pattern is not presented for a long period of time. During that time, random spikes will arrive. Because they all arrive after the last time the neuron fired (during a pattern), they are all "too late", and would cause LTD, were it not for the limited time window of STDP. Because of the window, as long as a neuron doesn't fire too often outside of its pattern, it will not forget.

Masquelier et al. [11] mentioned that in their study, a neuron almost never became selective to two different patterns, unless the patterns (or parts of them) were very similar. However, in the second experiment in this paper, a neuron quite frequently ended up firing during two patterns. Apparently, these neurons found parts of patterns that were similar enough to learn to fire upon, while keeping weights low enough so as not to fire too often during random input. They did however have slightly larger false positive rates than the neurons that detected only one pattern. The reason for this difference between the studies might be that, because only 100 inputs are used here, instead of 1000 in the

study by Masquelier et al., (parts of) the patterns used here are more likely to be similar to each other.

In the second experiment, the output neurons inhibited each other directly. However, it is also possible for them to excite an inhibitory neuron which in turn inhibits the other output neurons. I found this to work, but because of the software used here, it adds at least another millisecond of delay between the firing of an output neuron and the arrival of the inhibiting spike at the other output neurons. This decreases the effectiveness of this method of competition between output neurons.

In the third experiment, we sensitized a neuron to a pattern by setting its delays to certain values. It should also be possible to sensitize a neuron to a certain pattern by pre-setting the weights of its connections. One might call the first method temporal sensitization, because the neuron is made sensitive to patterns with certain spike timings, and the second method spatial sensitization, because the neuron is made sensitive to patterns with firings on certain neurons. However, in the case of patterns that are spatially similar (they fire upon the same neurons), spatial sensitization would of course have little effect.

4.1 Future work

The time resolution of 1 ms caused some problems in the second experiment. Another small issue with the time step of 1 ms is that some neuron models (not the ones used here) seemed to be able to fire multiple times within one millisecond. By the current software this would be handled as one firing during that millisecond. Therefore, for future simulation studies, a higher time resolution, or event-driven software with continuous time, might be preferred.

Here, for STDP I took into account only the last firing time of a neuron, and the last spike arrival time. As also mentioned by Masquelier et al. [10] it would be possible to look at triplets of spikes. With spike triplets, two pre- and one postsynaptic spike, or two post- and one presynaptic spike are taken into account when deciding what the weight change should be. This may have a positive or negative effect on pattern recognition performance. I am also curious to see what effect the addition of short-term synaptic plasticity would have on performance. As was also mentioned in the introduction, in future simulation studies, STDP could be modulated (with reward or punishment), to allow for the learning of some patterns, and not others.

The network configuration used here was very simple: several input neurons were connected to one or more output neurons. If this setup would be layered, that is, the "output" neurons were connected to other final output neurons, this would allow longer patterns to be detected by one final output neuron. Other changes to the network structure could also be investigated in future work.

Axonal delays have already been found to be used in the brains of barn owls, for sound localization. New research may find other areas in brains where axonal delays play an important role.

If, as simulated in the third experiment, neurons in the brain are sensitive to certain spike patterns due to the delays of their connections, it becomes interesting to know how these delay configurations are formed. Maybe they are mostly static, and selected by evolution. Perhaps there are mechanisms in the brain that allow delays to be adjusted, or selected. Or possibly, due to recurrent connections, there are so many connections with different delays between

neurons, that all possible patterns are already represented in the structure of the brain, just waiting to be activated.

References

[1] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J.M. Bower, M. Diesmann, A. Morrison, P.H. Goodman, F.C. Harris, et al. Simulation of networks of spiking neurons: a review of tools and strategies. *Journal of computational neuroscience*, 23(3):349–398, 2007.

[2] N. Caporale and Y. Dan. Spike Timing–Dependent Plasticity: A Hebbian Learning Rule. 2008.

[3] R.V. Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502, 2007.

[4] D.O. Hebb. The organization of behavior: A neuropsychological approach. *NewYork: John Wiley & Sons. Hinton, GE (1989). Deterministic Boltzmann learning performs steepest descent in weightspace. Neural Computation*, 1:143–150, 1949.

[5] EM Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[6] EM Izhikevich. Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, 15(5):1063–1070, 2004.

[7] E.M. Izhikevich. Polychronization: Computation with spikes. *Neural Computation*, 18(2):245–282, 2006.

[8] E.M. Izhikevich. Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral Cortex*, 2007.

[9] W. Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[10] T. Masquelier, R. Guyonneau, and S.J. Thorpe. Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLoS ONE*, 3(1), 2008.

[11] T. Masquelier, R. Guyonneau, and S.J. Thorpe. Competitive STDP-based spike pattern learning. *Neural computation*, 21(5):1259–1276, 2009.

[12] B. Nessler, M. Pfeiffer, and W. Maass. STDP enables spiking neurons to detect hidden causes of their inputs. 2009.