Temporal Sequence Learning in Spiking Neural Networks

Belal Mandurah

MCIS

2011

Temporal Sequence Learning in Spiking Neural Networks

Belal Mandurah

A dissertation submitted to

Auckland University of Technology

in partial fulfilment of the requirements for the degree of

Master of Computer and Information Science (MCIS)

2011

School of Computing and Mathematical Sciences

Faculty of Design and Creative Technologies

Knowledge Engineering and Discovery Research Institute

(KEDRI)

Supervisor: Dr. Ammar Mohemmed

Table of Contents

Table of Figures:

Table of Tables:

Attestation of Authorship:

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgments), nor material which to a substantial extent has been submitted for award of any other degree or diploma of a university or other institution of higher learning.

Acknowledgments:

I received great assistance from my supervisor Dr. Ammar Mohemmed. He provided me with the main code used in the experiments and helped me with adjusting the code. He also provided me with related research sources papers to guide me in this dissertation. I have learned greatly with him and I can’t thank him enough for what he did for me. I also would like to acknowledge the great job of Ms. Catriona Carruthers in proofreading this dissertation.

Abstract:

Spiking Neural Networks (SNN) are the third generation of artificial neural network (ANN). Like the brain’s neurons, they use spikes (pulses) to propagate information. Spike sequence learning has many applications for example in speech recognition and motor control. One of the main issues for sequence generation is learning. There are two main types of learning, unsupervised learning and supervised learning. Supervised learning, like Back Propagation (BP) is based on using a teacher signal to tune the connection weights to guide the network to produce a desired output for a specific input.

In this work, two supervised sequence learning schemes will be investigated. The first scheme uses particle swarm optimization (PSO). Using PSO, the SNN consists of multiple layers of neurons connected by dynamic synapses to increase its computation power. Due to the limitation in scalability of PSO, the second algorithm, SPAN, will be investigated which is able to use spatial temporal data. SPAN uses a simpler architecture; the network consists of a single neuron with multiple synapses. Though it originally uses static synapses, the synapses will be replaced with dynamic synapses to examine the impacts, if any exist, on the network and on learning performance.

A performance evaluation of the two methods, using different configuration for the parameters of the dynamic synapses and using different input data train, will be undertaken, as well as an evaluation of SPAN when the synapses are replaced with dynamic ones.

The main research question is how the dynamic synapses affect the learning and performance of the two above learning schemes. The sequence learning algorithm with PSO proved more complicated to optimize several parameters than optimizing one parameter per synapse with SPAN. Learning multiple inputs using SPAN with dynamic synapses proved faster than using static synapses. Nonetheless, memorizing the sequences using SPAN with the dynamic synapses did not show any improvement. SPAN has been proven to be able to successfully learn and classify multiple sequence trains with a simpler model and with the fine tuning of one parameter per synapse. This is a very interesting area to work in, and further future work can be done to improve each system.

CHAPTER 1

1. Introduction:

Understanding and simulating the learning process occurring in the brain is currently a major research area and many interesting new directions have appeared recently. Especially exciting are the engineering applications that may developed on the basis of such brain-like information processing algorithms. The idea of the dissertation is to investigate recently presented learning algorithms for Spiking Neural Networks (SNN) regarding its performance when additional, more biological plausible components, i.e. dynamic synapses, are introduced into the training process.

After the latest neurological studies, researchers developed the SNN and it is the most biologically plausible and accurate network to process neural information of all previous generations of artificial neural networks (Paugam-Moisy, 2006). Sequence learning with SNN can be implemented to solve complex real world problems and applications. It can be used, for example, in speech recognition, DNA sequences, and time series prediction (Sun & Giles, 2001, Tsodyks, Pawelzik, & Markram, 1998). Though sequence events differ from one application to another, it is important to know the order of events and, in other tasks, the timing of the event is more crucial (Sichtig, 2007).

In this dissertation, the main objective is to investigate supervised sequence generation learning with dynamic synapses. The dissertation starts with a literature review of the related work. This will cover related research that has used learning algorithms for spiking neural network with its applications. Then, the paper discusses the issues and limitations of SNN. After that, a spiking neuron model called Integrate and Fire is explained. There are several types of integrate and fire models model such as the Leaky Integrate and Fire model and the Izhikevich neuron model, and we explain why the former model was chosen. The chapter ends with reviewing and explaining dynamic synapses model, its formula and its main parameters.

Then, chapter three explains the learning scheme and algorithm used in the experiment for sequence learning, mainly PSO and with fitness function. PSO is used to tune three parameters of the dynamic synapses. Namely the recovery time, the facilitating time and the synaptic efficacy. This scheme learns the target train and generates an output train similar to the given target train. It continually compares the output with the target train in every iteration process, using the similarity measure neuron, and then modifies the parameter of the synapses by using PSO. The chapter then describes the experiment carried out, including the data, parameters, setup and architecture of the network. We experiment with this algorithm to assess the performance of the learning scheme. The last part of the experiment of this chapter deals with analysis of the results.

In chapter four, SPAN is the second investigated sequence learning algorithm. Unlike with PSO, SPAN modifies only the synaptic weight of the network iteratively to produce a desired output train. It can receive multiple inputs and learn spike trains to generate an output as similar to the target train by adjusting the synaptic weights of the network after comparing the differences between the output and the target train when the spike trains are convolved with a kernel function. However, this original SPAN uses static synapses and we need to observe if there is any difference to performance or in computational power if static synapses are replaced with dynamic ones.

In this SPAN chapter, there will be two different experiments. The first experiment's aim is to evaluate the ability and performance of SPAN in learning multiple inputs patterns when using dynamic synapses and comparing it to SPAN using static synapses. This experiment will demonstrate the robustness of the method when dynamic synapses are used, which is important for real world applications. The second experiment's purpose is also evaluating the ability and performance of SPAN to classify and memorize the trains with different classes when the synapses are replaced with dynamic synapses and comparing these results with SPAN using static synapses. This experiment demonstrates how many patterns a single neuron can learn when dynamic synapses are used.

The hypothesis is that since dynamic synapses are more biologically plausible than the static synapses, then the dynamic synapses will perform better in the experiments.

Finally, the last chapter ends the dissertation with a conclusion and future directions.

CHAPTER 2

2. Literature Review: SNN for Spike Sequence learning

In this second chapter of the paper, related research papers in the area of sequence learning with spiking neural network will be reviewed first. Then, the mathematical models and the computational units from the neural systems, which are used in the experiments, are divided into sections and explained.

2.1 Related literature:

In machine learning there are several algorithm types for learning, classifying and clustering data: supervised, unsupervised or semi-supervised learning. The supervised learning method generates the output when given a sample. In this case, we need an input value and a desired value to generate the output similar to the desired example. The unsupervised method discovers and finds hidden structure in unlabelled data. The semi-supervised learning method combines the labelled and the unlabelled structure to generate an appropriate function or outcome. The unsupervised method is widely investigated, but it is not appropriate for learning tasks that are to be used for a specific goal (Ponulak & Kasiński, 2010). We will not go into more detail about these other methods because they are out of scope of this dissertation. Thus, we will mainly discuss the supervised methods used with SNN and they are shown briefly in Table 1.

Supervised learning algorithms for SNN have been proposed in different research papers over the last decade. For instance, Spike-Prob, Tempotron ReSuMe and Chronotron are supervised learning algorithms used with spiking neural networks. First on the list, Spike-Prob, presented by Bohte, Kok, & La Poutré, (2000), is a supervised learning method for SNN based on precise spike time encoding. It is tested on classification problems such as extended XOR classification problem and real-world benchmark problems, such as the Iris, Wisconsin Breast Cancer and Statlog Landsat datasets (Bohte, Kok, & La Poutré, 2000). This model configuration has a multi-layer feed forward network with an error back propagation algorithm. The Spike-Prob uses a “gradient descent approach” (Mohemmed, Schliebs, Matsuda, & Kasabov, 2011) which adjusts the network synaptic weights to obtain a spike at the desired time. It assumes that each neuron is allowed to fire only once during a single simulation cycle Spikes fired at a specific time encode specific information. If a spike train has more than one spike, then Spike-Prob cannot be trained to generate the desired train of spikes (Mohemmed, Schliebs, Matsuda, Kasabov, 2011).

Tempotron is a neuron model proposed by Gütig & Sompolinsky (2006) that consists of a leaky integrate and fire neuron for classifying the input information received, including category information that is not encoded in spike counts. Tempotron also uses the dynamics of a gradient descent approach, and “poorly classify spike patterns that depends on the basis of temporal features that extend beyond a single integration time” (Gütig & Sompolinsky, 2006), limiting Tempotron to learn temporally localised data separated by long time periods. Thus, it is not capable of carrying out specific tasks that carry additional temporal data information, which makes it only suitable for binary classification problems (Gütig & Sompolinsky, 2006). Solving this requires more additional work on memory mechanisms or on slow synaptic dynamics and possibly having a multilayer architecture (Gütig & Sompolinsky, 2006).

The Remote Supervised Method (ReSuMe) is a set of Hebbian-based supervised learning algorithms presented by Ponulak & Kasinski (2006). ReSuMe uses a function called learning window. This rule is used with spike time dependent plasticity (STDP) and with anti-STDP (Gerstner & Kistler, 2002). This learning window concept uses two rules, the presynaptic and reference spike time rule or pre-and postsynaptic spike times rule, to obtain the desired spike target train with very high precision (Ponulak & Kasinski, 2006). ReSuMe has been effective when used with liquid state machine networks. However, it only tunes one parameter, which is the synaptic weight.

Another learning method was proposed by Florian (2010) and called Chronotron. Chronotron uses two versions of learning rules. By using a similarity measure and by optimizing the parameters of the neurons and the synaptic weights, they minimize the error difference between the target train and the actual train (Florian, 2010). The first version called I-learning, and it is more biologically plausible but less efficient than the second version (Mohemmed, Schliebs, & Kasabov, 2011). The second version is E-learning, and uses the Victor-Purpura (VP) distance (Florian, 2010). The temporal data results for classification tasks were compared with ReSuMe and showed that Chronotron performed better (Mohemmed, Schliebs, & Kasabov, 2011).

Evolutionary algorithms (EA) were also used for training SNN for classification (Belatreche, Maguire, & McGinnity, 2006). They were used to tune the dynamic synapses parameter for a single feed forward network. Similarly, Pavlidis, Tasoulis, Plagianakos, Nikiforidis, & Vrahatis (2005) and Jin, Wen, & Sendhoff (2007) used EA to tune the synaptic weight and their connections. However, the methods were used on static data only and were not used with temporal sequence data.

Table 1 Related supervised learning algorithims for SNN to learn temporal sequence data

Supervised Algorithm	Approach used	Parameters tuned	Known Issues	Possible solutions
SpikeProb (S.M. Bohte et al., 2000)	Multi-layer feed forward network and gradient descent approach.	Synaptic weights.	Cannot be trained if more than one spike exists in the spike train.	---
Tempotron (Gütig & Sompolinsky, 2006)	A single neuron model and gradient descent approach.	Synaptic weights.	Spikes separated by long time periods are poorly classified.	More additional work on the memory mechanisms or on the slow synaptic dynamics and possibly having a multilayer architecture.
ReSuMe (Ponulak & Kasinski, 2006)	Learning window + STDP + Anti STDP.	Synaptic weight.	---	---
Chronotron (Florian, 2010)	A single neuron with a similarity measure.	Neuron parameters + synaptic weights.	---	---

2.2 Spiking neural networks (SNN):

SNN appeared in 1997 and became popular over the last fourteen years (Sichtig, 2007). SNN is considered to be the third generation of neural networks. Previous generations uses rate encoding for carrying information. Rate encoding carries information dependent on the frequency of the firing spikes. The rate of the action potential spikes increases if the stimulus of the neuron increases. While new neurophysiological studies showed that the precise time a neuron fires a spike carries further information (Sichtig, 2007). Combined with rate encoding, information is transferred by sending precise time sequenced spikes, which is called temporal encoding (Bohte, 2004). While previous generations of neural networks depended only on the rate of the spikes, having this new information SNN becomes more biologically plausible than previous generations (Sichtig, 2007). Figure 1 is an example of how rate encoding and temporal encoding fires spikes in 10 neurons.

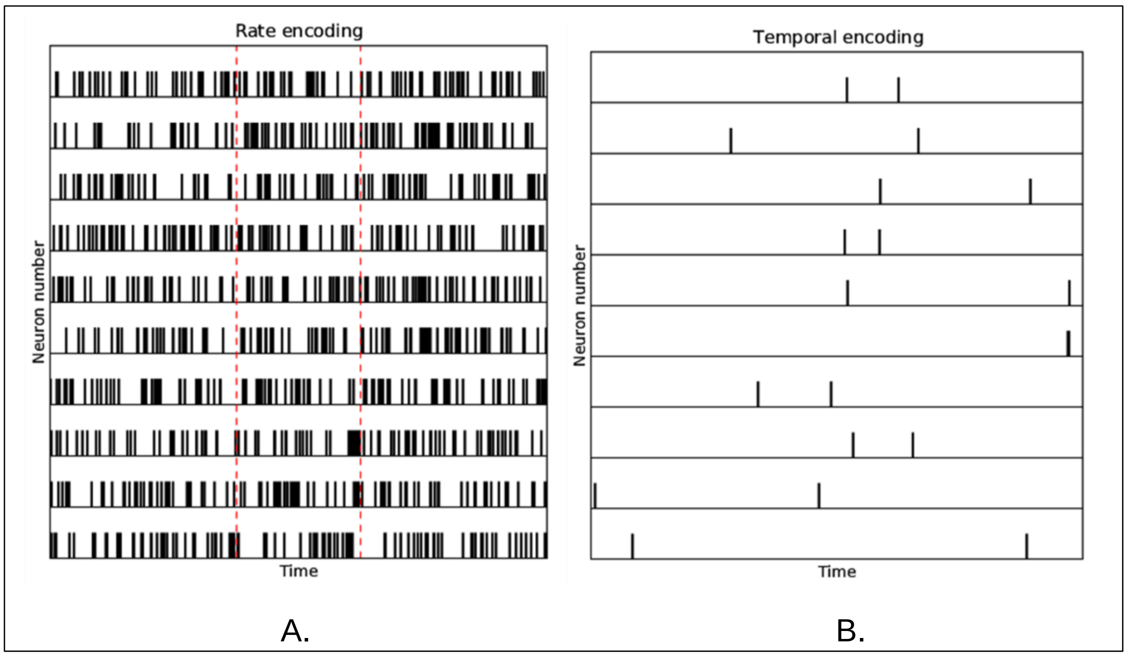


Figure 1 Rate encoding (A) where the frequency of the spikes are important and Temporal encoding (B) where the precise time of the spikes are important during an event.

Applications:

There are several successful applications for spiking neural networks which Bohte & Kok (2005) mentioned in their paper. It was reported that SNN effectively implements a complex temporal filter when the neurons are randomly connected (Bohte & Kok, 2005). For example, with a decoder, it can classify a temporally extended input, like speech recognition and time-series prediction (Sichtig, 2007). Learning and classification with SNN was also used in areas such as face recognition (Sichtig, 2007). Additionally, SNN can be used for robot control, vision and using scene segmentation for detecting criminal behaviour with surveillance cameras. There are also other learning applications, undertaken by previous generations of neural networks, was again implemented with spiking neural networks (Bohte & Kok, 2005; Sichtig, 2007).

There are four general goals that previous generations of neural networks aimed at: The first goal is auto association, where the network is used to reproduce the most similar pattern to a given training set pattern by tuning the parameters of the network. The second goal is pattern association, where the network processes an input data then produces an output consistent with its mapping, which was learned from a training set of pairs of patterns. The third general goal is classification and the final goal is clustering, where the network clusters the sets of data into groups or classes by discovering the features of the data without having a training set. (Sichtig, 2007)

Examples of applications that were undertaken by the traditional neural networks and were then solved with SNN were presented by Bohte & Kok (2005). SNN was reported as effective on the classification of Poisson spike train, and it was demonstrated on a temporal version of the classic XOR problem. It was reported that the XOR problem could be solved very easily in SNN without a hidden layer.

Moreover, Bohte & Kok (2005) reported that SNN could be used for associative memory for brain modelling and data retrieval in the form of the Willshaw model. It was posited that neural associative memories may have practical advantages over localized storage. Furthermore, SNN can be used as a function approximator. This might be seen as a classification problem. For example, SNN can interpolate between a set of data points, which approximates the function. It can be used with learning rules or algorithms to approximate the function. The researchers Bohte & Kok (2005) reported it was successfully used with semi-supervised and unsupervised datasets. Other advantages of having SNN with learning algorithms is it can use their inputs and project them into a high dimensional space to allow the readout function to be simple (Goodman & Ventura, 2006). Their readout function also has the ability to be memory less, meaning that with SNN, we depend on the network to remember and represent the past and current input simultaneously (Goodman & Ventura, 2006).

Issues and limitations:

Bohte & Kok in (2005) reported one issue with applications using spiking neural networks; they are “computationally more intensive than previous generations of neural networks”. However, they said that SNN reduces drastically the communication load between neurons and allows parallel implementations to be efficient. Sichtig (2007) also added some limitations of SNN. There is “limited empirical data and computational theory about computing time series in biological and artificial pulsed neural nets” (Sichtig, 2007).

The reason SNN was chosen:

The reasons why spiking neural network was chosen over previous generations was because, according to neurophysiological studies, SNN is conceptually state of the art, and represents the biological neural network more accurately than previous generations. In addition, SNN showed its effectiveness and accuracy in some experiments and applications with continuous data over the traditional neural networks (Sichtig, 2007). Traditional neural networks are most successful when processing static data (Sichtig, 2007).

2.3 Integrate and fire neuron model:

Integrate and Fire (I & F) is one of the possible models of a spiking neuron. The other two are the Spike Response model and the Hodgkin-Huxley model (Paugam-Moisy, 2006). Integrate and fire model have several model types and they are based on an electric circuit, having a capacitor (C) and a resistor (R) (Paugam-Moisy & Bohte, 2009).The first model explained is the Leaky Integrate and Fire (LIF) neuron, which is a simplification of the Hodgkin-Huxley model. This model can be defined by the following equation:

$$\tau_m \frac{dv}{dt} = -v + RI(t) \tag{1}$$

Where $\tau_m = CR$ is the time constant of the neuron membrane, R is the resistance of the membrane and $I(t)$ is the input current. The neuron generates a spike when the membrane voltage reaches a threshold value. A threshold value is reached when a sufficient number of spikes have occurred. After reaching the threshold, the potential voltage v drops to a value of V_{reset} and stays at that level for a period of τ_{ref}, which is called the refractory period. All incoming spikes are ignored during the refractory period. In addition, the membrane voltage is resting at a value of V_{rest} until the neuron activates again.

Integrate and fire neuron models consider every spike as a uniform event defined only by the time it fires and neglects the shape of the action potentials or spike or pulse. Each neuron can simulate either an “integrator” or a “resonator” by changing the parameters but cannot be both at the same time as their properties are mutually exclusive (Paugam-Moisy, 2006).

The second integrate and fire model is the Izhikevich neuron model. The authors Sichtig (2007) and Paugam-Moisy and Bohte (2009) reported that the two dimensional Izhikevich neuron model is balanced between the biophysical plausibility and versatility of the Hodgkin-Huxley type model and the computational efficiency or cost of the integrate and fire and resonate and fire type models. By using Bifurcation methodologies, the author of the model Izhikevich (2003) was able to reduce many Hodgkin-Huxley type models to a two dimensional system. The Izhikevich neuron model is defined by the coupled equations:

$$\frac{du}{dt} = 0.04v(t^2) + 5v(t) + 140 - v(t) + I(t) \tag{2}$$

$$\frac{dv}{dt} = a(bu(t) - v(t)) \tag{3}$$

And after spike resetting: if $u \geq \vartheta$ then $c \rightarrow u$ and $v + d \rightarrow v$
where u is the membrane potential of the neuron, I is the synaptic current, t is the time and v is the membrane recovery variable. While a, b, C and d are the four dimensionless parameters that can be tweaked for achieving the desired spike behaviour (Izhikevich, 2003). The spike behaviours are such as, fast spiking, regular spiking, resonator, chattering, intrinsically bursting, thalamo cortical and low threshold spiking. The parameter a represents the time scale of the recovery variable v, the smaller the value of a the slower the recovery. The parameter b is the sensitivity of the recovery variable u to the subthreshold fluctuations of the membrane potential. Great values of b leads u and v to stronger posibilites for subthreshold oscillations and low threshold spiking dynamics. The parameter c is the after spike reset value of the membrane potential caused by a fast high threshold. Finaly, the parameter d describes the after spike reset of the recovery variable caused by slow high threshold.

The last two models are the Quadratic Integrate and Fire (QIF) model and the Theta Neuron model (Paugam-Moisy & Bohte, 2009). (There are also other existing spiking neuron models such as the gIF model, which is an intermediate model, but they will not be covered in this paper.) The QIF is simpler to understand and less complex than other models and it also realistically captures the behaviour of the biological neuron. The Hodgkin-Huxley models are the most realistic to a biological neuron, while the LIF is a simplification and does display many of the important properties of biological spiking neurons (Paugam-Moisy & Bohte, 2009). However, the computational costs of these models are different. For example, the Hodgkin-Huxley model requires twelve thousand floating point operations (FLOP). The LIF model has five floating point operations and the Izhikevich model has thirteen floating point operations

(Paugam-Moisy & Bohte, 2009). For simplicity and cost effectiveness, the LIF model was chosen in our experiments over the other models.

2.4 Dynamic synapses:

The dynamic synapses' strength relies on constantly changing its parameters according to the temporal pattern. Dynamic synapses have been successfully used in several applications such as speech recognition (Jim-Shih Liaw & Berger, 1997). They have been also used as computational model to behave like finite state machines, which are computer science models for computational time series, and have yielded good performances compared with other artificial networks that do not have any biological realism (Natschläger & Maass, 2002).

Dynamic synapses "control how the pattern of amplitudes of post-synaptic responses changes with the temporal pattern of the pre-synaptic" (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011). Having this information, several phenomenological models have been proposed that behave similarly to the dynamics of biological synapses. One model proposed by Tsodyks et al. (1998) explains that the postsynaptic responses are generated by either facilitating or depressing. The characteristic of this model's synapses is having a finite amount of resources, and the presynaptic spike activates a fraction of (U_{SE}), which is the utilization of the synaptic efficacy, when arriving at a time (t_{sp}) and then inactivating quickly with a time constant (t_{in}), which is about of few milliseconds. After that, it recovers with a time constant (τ_{rec}), which is about 1 second. The model of Tsodyks et al.'s (1998) is defined by the equations:

$$\frac{dx}{dt} = \frac{z}{\tau_{rec}} - U_{SE}x(t_{sp} - 0)\delta(t - t_{sp}) \tag{4}$$

$$\frac{dy}{dt} = -\frac{z}{t_{in}} + U_{SE}x(t_{sp} - 0)\delta(t - t_{sp}) \tag{5}$$

$$\frac{dz}{dt} = \frac{y}{\tau_{rec}} - \frac{z}{\tau_{rec}} \tag{6}$$

The Symbols x, y and z are the fractions of resources. Where x is for recovery state, y is for the active state and z is for the inactive state. Delta $\delta(x)$ is the

Dirac delta function. There are two major parameters for the model, the utilization of the synaptic efficacy U_{SE}, which determines the dynamic of the synaptic response, and the absolute synaptic strength A_{SE}, which is revealed only when all resources are activated (Tsodyks et al., 1998).

However, the equations (4), (5) and (6) do not include a facilitating mechanism, which is important in synapses between pyramidal neurons and inhibitory interneurons (Tsodyks et al., 1998). Facilitation and depression are two synaptic plasticity processes, and according to each process, the parameter synaptic efficiency, or what is called the weight, changes dynamically with each pre-synaptic spike. To add facilitation to the equations, the value of U_{SE} is assumed not to be fixed and rather to be increased in each presynaptic spike by a certain amount (Tsodyks et al., 1998). The U_{SE} value refers to u at the facilitating time (τ_{fac}) and U_{SE} is equal to U_0 at the time of the first spike. The new equations are formed as the follows:

$$\frac{dx}{dt} = \frac{z}{t_{rec}} - ux\delta\left(t - t_{sp}\right) \tag{7}$$

$$\frac{dy}{dt} = -\frac{y}{t_{in}} + ux\delta\left(t - t_{sp}\right) \tag{8}$$

$$\frac{dz}{dt} = \frac{y}{\tau_{rec}} - \frac{z}{\tau_{rec}} \tag{9}$$

$$\frac{du}{dt} = -\frac{u}{\tau_{fac}} + U_0(1-u)\delta(t - t_{sp}) \tag{10}$$

Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov (2011) mentioned that depressing and facilitating mechanisms are intricately interconnected. It is observed that stronger facilitation leads to higher values of u, which leads to stronger depression (Tsodyks et al., 1998). In other words, when firing at a low rate, the spike signals when depressing are different from the facilitating spike signals. However, if the firing rate increases, the depressant spike signals become more similar to the facilitating spike signals (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011).

As mentioned previously, the parameters of the dynamic synapses U_{SE}, τ_{rec} and τ_{fac} are the main characteristics of dynamic synapses and they can be tuned. In Figure 2, the membrane voltage of a post synaptic neuron receiving from the pre synaptic over a dynamic synapse is illustrated with different dynamic synapse configurations, while the vertical green lines represent the input spikes. These were simulated and illustrated using the NEST simulator by Gewaltig & Diesmann (2007).

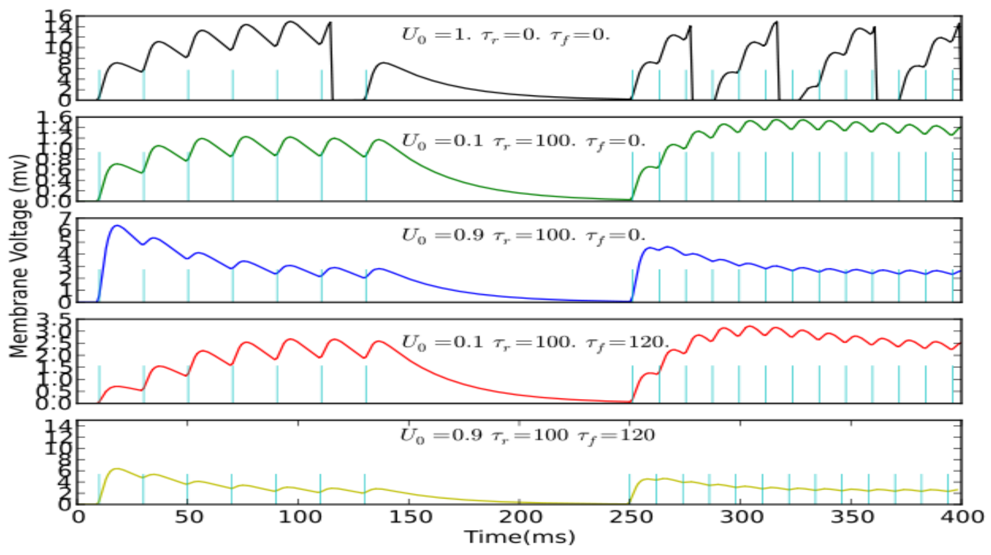


Figure 2 This is the membrane voltage of the post synaptic neuron after receiving spikes over dynamic synapses. Each plot has a different configuration of the parameters U_{SE}, τ_{rec} and τ_{fac} (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011).

CHAPTER 3

3. Optimizing SNN for Sequence learning using Particle Swarm Optimization (PSO):

In this chapter, the general concept of PSO and its history will be reviewed first. Then, the mathematical equations and the fitness function, which are used in the experiments, are divided into three sections and explained in more detail. After that, the experimental work is reviewed and the results are analysed.

3.1 General principles:

PSO uses a population of particles. Each particle has a position and a velocity in space, (which is why it is more appropriate to call them particles rather than calling points), and moves around in order to find the solution for the problem being solved (Chen & Yu, 2005). PSO is chosen because it is believed to be simple, effective, and versatile and emulates nature rather than trying to control it. Particle swarm optimization has a wide range of applications and it has proved simple and robust (Kennedy & Eberhart, 1995). In addition, Kennedy & Eberhart (1995) stated that the code for PSO is inexpensive and requires only basic computational operators as will be seen in this paper. There are two concept methodologies for particle swarm optimization. According to Kennedy & Ebhart (1995), the first methodology is related to artificial life in general, bird flocking, herd behaviour, fish schooling, and human behaviour. The second methodology reported has links to genetic algorithms and evolutionary programming.

The development of particle swarm optimization was motivated by other scientists trying to simulate the social behaviour observed in nature and moving organisms (Kennedy & Ebhart, 1995). Specifically, when large numbers of birds synchronise and flock together, changing direction suddenly, scattering, regrouping and also social behaviour observed in human. Humans are more complex than animals. They adjust their physical movements and also their cognitive or experimental variables as well. Humans adapt to their social peers by adjusting their beliefs and attitude (Kennedy & Ebhart, 1995). The overweighting advantages of this behaviour suggest a hypothesis, which was fundamental in the development of particle swarm optimization. The hypothesis the authors posited was “that social sharing of information among conspeciates offers an evolutionary advantage” (Kennedy & Ebhart, 1995). Thus the beginning of PSO algorithm history started with simulating this simple social environment and plotting the “graceful unpredictable” choreography of flock of birds.

Kennedy & Ebhart (1995) described the precursors of the etiology of particle swarm optimization. One of the precursors is the multidimensional search. This algorithm is suitable for a social behaviour model which is a multidimensional and collision free. An experiment was performed by Kennedy & Ebhart (1995) on a three layer neural network for solving an XOR problem. In this experiment, the weights of the neural network were adjusted using PSO before they were trained and the results are reported to have been decent.

PSO was also used to train neural networks for classification (Carvalho & Ludermir, 2006). For example, classifying the Fisher Iris set (Kennedy & Ebhart, 1995) and, in the medical field, problems such as cancer, diabetes and heart conditions (Carvalho & Ludermir, 2006). Kennedy & Ebhart (1995) stated that PSO train neural networks as effectively as the error backpropagation method. Another application is optimizing a difficult function. An extremely nonlinear Schaffer f6 function was optimized when using PSO by finding the global optimum in each run (Carlisle & Dozier, 2001, Kennedy & Eberhart, 1995). PSO was reported as an effective method in terms of the number of evaluations required to reach a certain performance level similar to elementary genetic algorithms effectiveness (Kennedy & Eberhart, 1995). To explain PSO mathematically, PSO behaves as a “swarm of particles or agents that search for an optimum position or solution” (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011) and must have the following rules:

$$vel_i(t+1) = \alpha vel_i(t) + c_1 r_1 \left(P_i - x_i(t)\right) + c_2 r_2 \left(Pg - pos_i(t)\right) \tag{11}$$

$$pos_i\,(t+1) = pos_1\,(t) + vel_i\,(t+1) \tag{12}$$

Within the range of [$pos^{\min}$,$pos^{\max}$] and [vel^{min}, vel^{max}], the values of the position pos_i and the velocity vel_i are both randomly selected. The others r_1 and r_2 are random numbers in the range of [0, 1] while c_1 and c_2 are two constants, which control the influence of P_i and P_g. The symbol P_i is the best position chosen by the particle, which produces the best fitness value, and P_g is the best position chosen by all the particles. Finally, the last symbol α is the inertia weight of PSO. This inertia weight controls the impact of the previous velocity of the particle on its current one. (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011)

Particle swarm optimization is highly dependent on random processes and uses the concept of fitness as an evolutionary computation (Kennedy & Eberhart, 1995). PSO and genetic algorithms share similar conceptual operations. Specifically, the adjustment of the best positions of the particles in PSO made PSO conceptually similar to the adjustment of crossover utilization in genetic algorithms (Kennedy & Eberhart, 1995).

In spiking neural networks with particle swarm optimization for sequence learning in our experiment, PSO is used to adjust the parameters of the dynamic synapses, which will in return generate the target spike sequence. The dynamic parameters of the spiking neural network's synapses that will be adjusted with PSO are U_{SE}, τ_{rec} and τ_{fac}, which were explained in the previous section in equation (7), (8), (9) and (10). The next step is encoding these parameters. After that, a fitness function must be applied to measure the differences between the actual output sequence and the target spike sequence. (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011)

3.2 Fitness functions:

Fitness function means measuring the differences and computing the relationship between the given target sequence (spike train) and the output sequence. The similarity measures are important for classification, clustering or for any form of spike analysis (Paiva, Park, & Príncipe, 2009). In our experiment for example, the fitness function is used by PSO for optimizing the dynamic parameters of the spiking neuron network to generate an output similar to the desired target sequence. There are several prior studies and proposed fitness functions.

The researchers, Dauwels, Vialatte, Weber, & Cichocki (2009) and Paiva et al. (2009) investigated and mentioned several similarity measures in their research papers. These similarity measure methods can be categorised into two groups (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011). The first group consists of the binning base measures and the second group consists of the binless measures.

Binning measures means dividing the sequence/spike train into bins and measuring its cross correlation to compute similarity. These following similarity measures are defined as binning base measures and were discussed in Dauwels et al.'s (2009) paper. For example, the Vector-Purpura distance metric, the van Rossum distance metric, the Schreiber et al. similarity measure, the Hunter-Milton similarity measure, event synchronization by Quiroga, and the stochastic event synchrony measures (SES) by (Dauwels et al., 2009). Nevertheless, there is an issue with all binning measures. That is, they miss all the temporal structure taken in the spike train (van Rossum, 2001). Other difficulties associated with binning when using small bin sizes, the quantization of the spike times leads to boundary effects (Paiva et al., 2009). Small bin sizes also cause estimation problems, needing longer averaging windows in a stationary situation (Paiva et al., 2009).

Several binless measures have been proposed to avoid the limitations and difficulties of binning measures (Paiva et al., 2009). For example, the Victor-Purpura's (VP) distance, the van Rossum's distance, the correlation based measure, the inter-spike interval (ISI) distance, the reliability measure, the metric by Houghton generalizing van Rossum's distance and the metric generalizing the VP to simultaneously measure the distance between spike trains (Paiva et al., 2009). The VP distance defines the distance between spikes as the cost. It differentiates one spike from another by either the insertion/deletion of a spike or the shifting of the spike in time. It calculates the cost when one spike train transforms into another spike train. Nevertheless, it is difficult to determine the inserted/deleted spike if the two spike trains have an unequal number of spikes (van Rossum, 2001).

The van Rossum's distance is one binless measure that takes the temporal structure of spike trains into account. This measure is closely related to the VP distance (Paiva et al., 2009). The van Rossum's distance is defined by making the following measures. First, the spike trains are converted into continuous time signals. This is done by convolving each spike with an exponential function. Mathematically, the spike train is defined by this equation:

$$S(t) = \sum_{sp}^{M} \delta(t - t_{sp}) \tag{13}$$

Where t_{sp} is the time of the arrival of the spike and $t_{sp} > 0$. When changing the function into an exponential function, it will be adding an exponential tail to all spikes. This following equation is the replacement exponential function:

$$S^c(t) = \sum_{sp}^{M} H\left(t - t_{sp}\right) e^{\frac{-(t-t_{sp})}{\tau_c}} \tag{14}$$

In the previous function (14), (τ_c) is the time constant of the exponential function and (H) is the Heaviside step function:

$$H(x) = 0 \text{ if } x < 0 \text{ and } H(x) = 1 \text{ if } x \geq 0 \tag{15}$$

Computing the distance between two spike trains is the next step. For example, if $f^c(t)$ and $g^c(t)$ are two spike trains, the distance is:

$$D^2(f^c, g^c)_{\tau_c} = \frac{1}{\tau_c} \int_0^{\infty} [f^c(t) - g^c(t)]^2 dt \tag{16}$$

A proposed fitness measure by Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov (2011) is a binless measure similar to the van Rossum's distance and avoids the difficulties of VP measures. They exploit a feature of a LIF neuron. The neuron changes its membrane potential with every incoming spike signal, which shows continuous representation of a spike sequence. For that, they used the neuron to measure the similarity between two or many spike sequences.

In Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov's (2011) paper, two synapses were connected to the LIF neuron. One of the synapses is excitatory and the other will be inhibitory. Excitatory means it increases the probability of an actual potential occurring and it is more likely for the target cell to fire while inhibitory means it is less probable for the target cell to fire. Each input spike sequence was connected to either one of the synapses. Then, the similarity was computed with the following integration function:

$$\mathrm{SM} = \int_0^L |S_d^c(t) - S_a^c(t)| dt \tag{17}$$

Where $(S_d(t))$ is the desired spike sequence and $(S_a(t))$ is the actual spike sequence. The $(S_d^c(t) - S_a^c(t))$ is proportional to the membrane voltage of the similarity measure, where c represents the continuous version of the sequence. Finally, L represents the simulation time. (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011)

When comparing spike sequences, the membrane time constant of the LIF neuron τ_m should be set appropriately (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011), and a trial and error method was used to set τ_m to have the best possible outcome. Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov (2011) specified that the parameter τ_m was the only parameter that could affect the similarity measure.

3.3 Experimental work:

The experiment in Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov (2011) work will be reinvestigated to verify the results, and exploiting any further information or exposing any issue with the methods that were used.

There is one main tool used in this research. All simulation experiments in this paper use NEST simulator (Gewaltig & Diesmann, 2007), which is written in Python programming language, for simulating the neuron networks. In Python, the Matplot library is used for drawing scientific plots and histograms. A description and further background of NEST and Matplot was taken from the Neural Simulation Technology (NEST) website and from the Matplotlib website.

3.3.1 The network architecture:

The network architecture used with PSO consists of three layers. Figure 3 below shows the approach used with a feed-forward network with a single hidden layer. The input neuron is connected to the hidden layer using the dynamic synapses. This hidden layer has a total of ten neurons, five neurons are inhibitory, and five neurons are excitatory. These neurons will be connected to an output neuron. Then, the output will be connected to the similarity measure neuron with inhibitory synapses. This division is done to provide biological plausibility which contain 80% inhibitory synapses and 20% excitatory synapses (Goodman & Ventura, 2006).

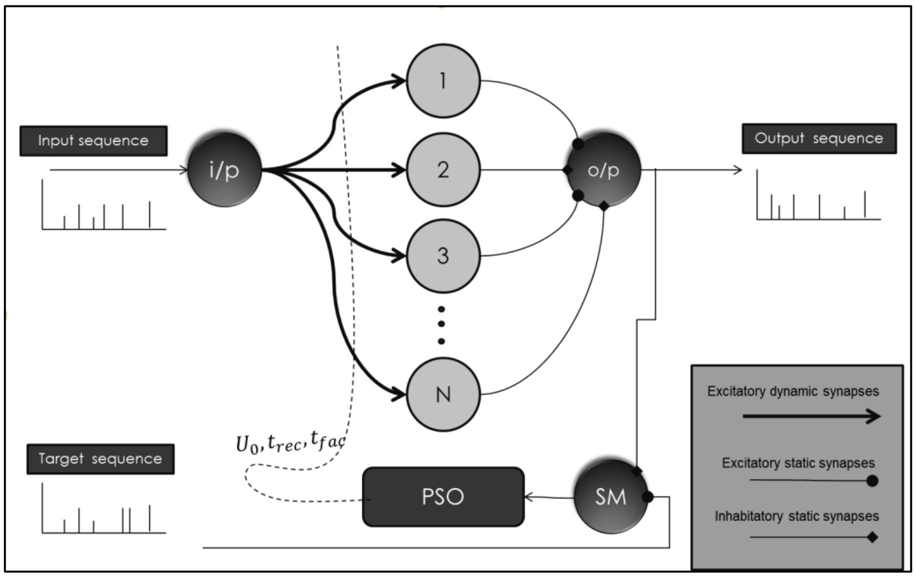


Figure 3 The network architecture for training the spiking neural network re-drawn from Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov (2011).

The similarity measure between the output sequence and the target sequence, which is connected using excitatory synapses, are computed with LIF neuron as a fitness function to PSO. These inhibitory and excitatory synapses are static and have a fixed weight that will not change during the simulation. After optimizing the parameters, PSO feeds back to the dynamic synapses of the hidden layer. (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011)

3.3.2 The experimental setup:

The configuration of the parameters of PSO and the LIF neuron in the simulation experiment are set as in the research of Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov (2011). The Table 2 below shows the configuration values which are used in the experiment simulation. In Table 2, the Maxiter is the maximum iterative process or epoch that can be used and R_{in} is the resistor. The number of particles used is 20 particles. The rest of the parameters were explained in the previous chapter.

Table 2 Parameter configration for PSO, Network and the SM neuron

PSO			
pos^{min}	{0.1, 1.0, 1.0}	pos^{max}	{0.9, 100.0, 120.0}
vel^{min}	{-0.01, -10.0, -10.0}	vel^{max}	{-0.01, -10.0, -10.0}
No.Particles	20	Maxiter	50
Network Neurons			
τ_m	10 ms	V_{rest}	0
V_0	7mv	R_{in}	1 GΩ
V_{reset}	0	-	-
SM Neuron			
τ_m	15 ms		

As mentioned previously, the method will be similar to the simulation experiment done by Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov (2011). Initially, we generated ten random settings for the dynamic synapses' parameters U_0, τ_{rec} and τ_{fac}. This is due to the fact that it is not possible with a single layer feed forward network to map the connection between the input sequence and the output sequence (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011). These ten randomly generated settings will generate ten random input sequences for each setting and will result in ten output sequences, therefore generating one hundred input and output sequence trains, and these trains will be used for performance evaluation.

In each example, there will be an input sequence that will be trained to generate the target sequence. Each example will be repeated ten times and each run will have different initialization settings for the PSO and for the neurons' parameters. These settings are generated randomly for each run. When the simulation or the training starts with a random setting for the dynamic synapses' parameters, PSO then is used to fine tune these parameters to generate the target sequences. The performance evaluation in this experiment is the ability of PSO to train the network so that it generates the target sequence. Having τ_m equal 15, computes SM for a value of 5.4 for each missing or extra spike difference, resulting in a value of one millisecond for each spike shift if SM is equal to 0.48 (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011).

3.3.3 Results and analysis:

First, the training result for the input train is shown in Figure 4. It shows the iteration number for one simulation and the SM value. The evolving output train is in red, which starts with only three spikes, reaching the target train in green. The SM value at the beginning is equal to 38.9, and, at the last iteration before reaching the target train, the SM value drops to 4.4, suggesting that the learning algorithm is able to train the network to produce the desired train.

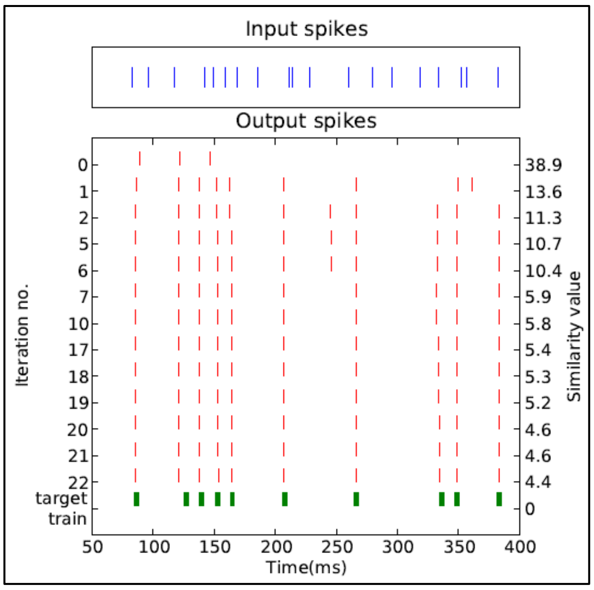


Figure 4 The output sequence while evolving to generate the target sequence (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011).

To see the effectiveness of the network when training 1000 experiments, we draw a histogram of the SM of the 1000 experiment before and after training. Histogram A in figure 5 is represents the SM values before training the network. After training the network with PSO, the outcome is produced and is shown in Histogram B. Figure 5.B shows that more than 85% of the 1000 experiments resulted in having a similarity measure value of less than five. This means that most of the results have a slight train shift when compared with the target train. The average value of SM for the 1000 experiment after training is 1.62, suggesting that the shift spike is approximately equivalent to 3.375 milliseconds. This experiment confirms the result of the experiment by Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov (2011). In addition, it demonstrates the effectiveness of the method when used on the temporal data. However, this method requires several parameters tuning, which is a more difficult task than adjusting a single parameter.

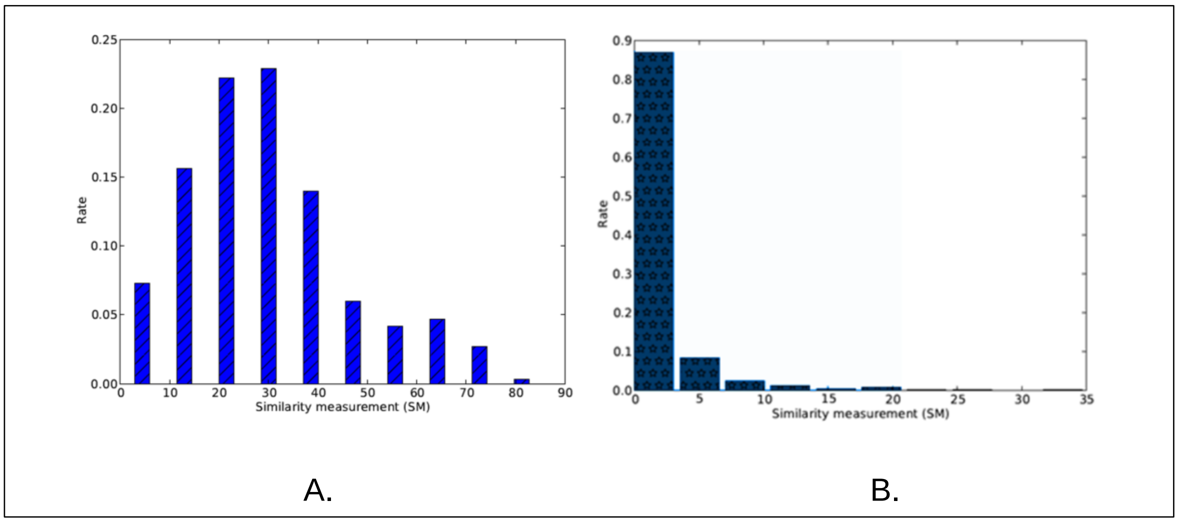


Figure 5 Histogram of the similarity measure before training (A) and after training the network (B) (Mohemmed, Matsuda, Schliebs, Dhoble & Kasabov, 2011).

CHAPTER 4

4. Sequence learning using Spike Pattern Association Neuron (SPAN) with dynamic synapses:

In this chapter, SPAN concept will be explained first along with its mathematical equations. After that, the conducted experimental work is described and the results are analysed.

4.1 Sequence learning using SPAN:

Spatial temporal pattern recognition gained interest in spiking neural networking in several pieces of research. Goodman & Ventura (2006) studied and used SNN with a supervised method in learning and recognizing spatial temporal patterns as liquid state machines (LSM) for solving real world problems such as, stockpile surveillance signal alignment and spoken phoneme recognition.

The researchers Mohemmed, Schliebs & Kasabov (2011) came up with a new supervised learning algorithm for spatial temporal information using one neuron and called it SPAN. This method is based on the Widrow-hoff or Delta rule (Mohemmed, Schliebs & Kasabov, 2011).The algorithm modifies the synaptic weights of the network iteratively to produce the desired output spike. It defines the error between the target train and the actual train by convolving each spike sequence with a kernel function (Mohemmed, Schliebs & Kasabov, 2011) . We will describe the synaptic and neural model first. Then the learning algorithm will be explained.

SPAN uses also the LIF neuron model for simulating the spiking neural network. The LIF neuron model has a synaptic current (I), which is modelled using an α-kernel. This is defined as:

$$I(t) = \sum_{j} w_{ij} \sum_{f} \alpha(t - t_j^{(f)}) \tag{18}$$

$$\alpha(t) = e\tau_s^{-1}\, t\, e^{\frac{-t}{\tau_s}}\, H(x) \tag{19}$$

Where w_{ij} is the synaptic weight describing the connection strength between neuron i and its pre-synaptic neuron j (Mohemmed, Schliebs & Kasabov, 2011), and $H(x)$ is the Heaviside function, which was described in equation (15). The algorithm of the researchers, Mohemmed, Schliebs & Kasabov (2011), starts with a Widrow-Hoff rule for adjusting the weight of the synapse i.

$$\Delta w_i^{WH} = \lambda x_i(y_d - y_{out}) \tag{20}$$

Where λ is a real-value positive learning rate, and x_i, y_d and y_{out} are the input train through the synapse i, the desired train and the actual network output respectively. The input spike sequences are convolved for the SNN similar to PSO. As for the Δw_i, it is obtained by integrating Δw_i^{WH} to update the weight of the synapse i.

$$\Delta w_i = \lambda \int x_i(t)(y_d - y_{out}(t))\, dt \tag{21}$$

The weights are updated in an iterative process called epochs (e) and all the training samples are presented sequentially to the system for each epoch (Mohemmed, Schliebs & Kasabov, 2011). After accumulating the computed Δw_i for each sample, the weights are updated to by using:

$$w_i(e+1) = w_i(e) + \Delta w_i \tag{22}$$

While the Error (E) is the area under the curve of the difference between the actual and the desired output $y_d(t) - y_{out}(t)$:

$$E = \int |y_d(t) - y_{out}(t)| dt \tag{23}$$

4.2 Experimental work:

In this section, the experiments done by Mohemmed, Schliebs & Kasabov (2011) for training a neuron for learning a spike pattern will be redone with static synapses and then with dynamic synapses. The main objective of the experiment is to compare the performance and efficiency of SPAN with static synapses to that with dynamic synapses. Different configurations of the dynamic parameters will be considered in order to study their impact on the learning. There are two types of experiments conducted in this section. In the first experiment, SPAN learning algorithm will be used to train a single LIF neuron to map a random input spike pattern to a specific target spike train.

The second experiment in this section is to test the memory capacity of the neuron in recognizing and memorizing different numbers of different input patterns using dynamic synapses with SPAN. In this second part of the experiment, the procedure is also similar to that experiment done by Mohemmed, Schliebs & Kasabov (2011). However, the synapse is changed to a dynamic synapse. Then, the results will be compared with the results obtained by Mohemmed, Schliebs and Kasabov (2011), where static synapses are used, and will also be compared with research results using an algorithm called the Chronotron learning method (Florian, 2010). Additionally, the load factor imposed by the task on the neuron is calculated and compared. This load factor is defined as the ratio of the number of input patterns per synapse ($\frac{p}{n}$).

Noting that Mohemmed, Schliebs & Kasabov (2011) reported in their experiment that increasing the number of synapses enables the neuron to recognize more patterns. The hypothesis in these following experiments is that, by using dynamic synapses we will obtain faster and better results than using static synapses, as dynamic synapses are more faithful to the biological synapse.

4.2.1 The network architecture:

The network structure of SPAN basically contains one neuron with n synapses. The architecture is able to receive spatiotemporal spike patterns. Each pattern has a number of spike trains equal to the number of synapses. The synapses are initially static, and then they are replaced by dynamic ones. The architecture is shown in Figure 6.

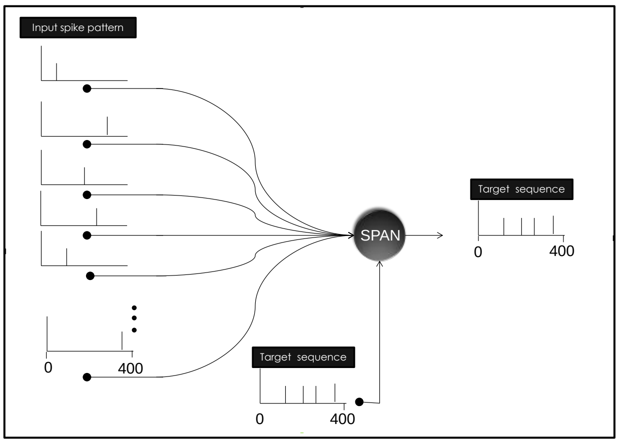


Figure 6 The SPAN architecture with 400 input synapses (redrawn from Mohemmed, Schliebs & Kasabov (2011)).

4.2.2 The experimental setup:

There are several values for the parameters for the dynamic synapses u, τ_{rec} and τ_{fac} in this experiment. The values of the dynamic synapses' parameters are shown in Table 3. As each parameter has four values, 64 different cases can be produced for the three parameters. These combinations are shown in Figure 7 and all are used in the experiment.

The configuration values for the rest of the experiment's parameters are fixed for both the static synapses and dynamic synapses and are shown in Table 4. In Table 4, Dt is the time resolution, c_m is the capacitor, Max_w is the neuron maximum weight, τ_m is the membrane time constant, τ_{ref} is the refractory period, V_{rest} is the voltage where spikes rest, V_{reset} is the voltage where spikes reset and V_th is the spike threshold.

Table 3 The possible values for u, τ_{rec} and τ_{fac}

u	τ_{rec}	τ_{fac}
0.25	200	50
0.50	400	100
0.75	800	200
1.0	1200	400

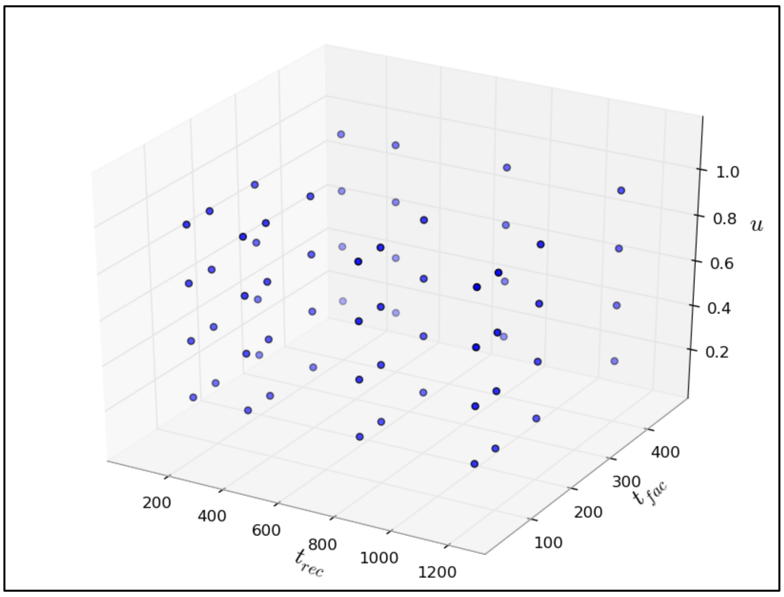


Figure 7 Each dot represents the setup value for the dynamic synapses parameters.

Table 4 The parameters' configuration for the simulation and for the neuron

Simulation							
Dt:	1.0	Seed:	1234	Accuracy th	0.0	Max epoch:	400
Sim time:	400.0 ms	Learning rate:	1.0	No jobs:	20		
Neuron							
τ_m:	10 ms	V_th:	0.0 mV	V_{rest}:	0 mV	τ_{ref}:	3.0 ms
c_m:	30pF	i_e:	0.	V_{reset}:	0.	Max_w:	10
Input							
No spike:	5	No classes:	1	No pat class	1	No inputs	400
Target train							
48.,55.,105.,115.,175.,205.,215.,249.,260.,270.,290.,325.,357.,370							

In the memory experiment, the maximum weight value is changed to a value of 2.5 for comparison with the results previously obtained using the static synapses, which it was set based on experimental observation according to Mohemmed, Schliebs & Kasabov (2011). The maximum epoch is changed to five hundred and the number of classes for the randomly generated input pattern is five (c=5). The target spike train is also changed, it emits spikes at times 33, 66, 99, 132 and 165. The synaptic weights were initialized randomly according to a uniform distribution and have a maximum value of 2.5 pA, which is based on experimental observation (Mohemmed, Schliebs, et al., 2011).

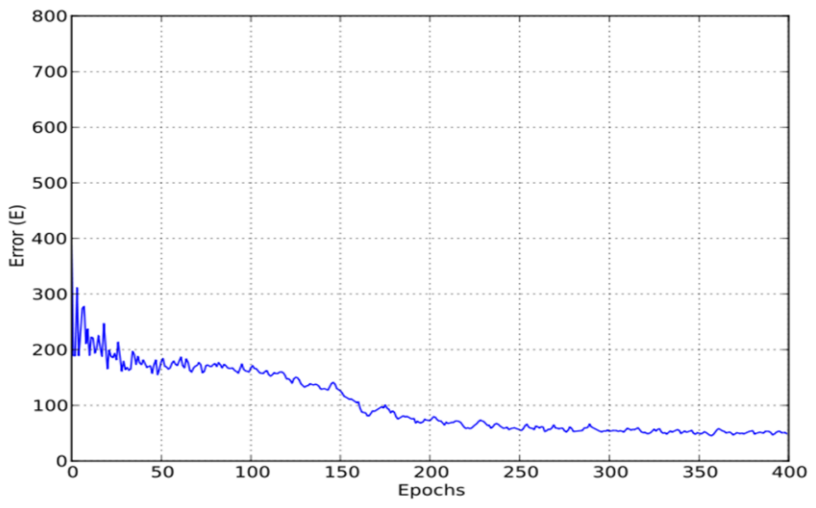
After explaining the setup configuration for our simulations, the experimental procedures are to follow. For the first experiment, initializing the synaptic weight is the first step. It is generated randomly between the range [0, 10pA] and assigned uniformly to all synapse. In addition, the five spikes' input patterns are generated randomly. For each of the sixty four configurations for the dynamic and static synapses, the model runs one hundred experiments, and runs for a maximum of four hundred epochs. Therefore, there are six thousand and five hundred trails. The average results for each of the dynamic synapses' configuration are plotted and the best result is used for comparison with the static synapses.

As for the memory capacity experiment, the best configuration resulting from the first experiment is used. Different values of input patterns (p) are generated randomly and assigned to the five different classes. The neuron is trained to fire a single spike at a specific time $t_d^{(i)}$, which is the time of either one of the target train spike times. According to Mohemmed, Schliebs & Kasabov (2011), the generated pattern is correctly classified if the corresponding output is within two milliseconds of the target train.

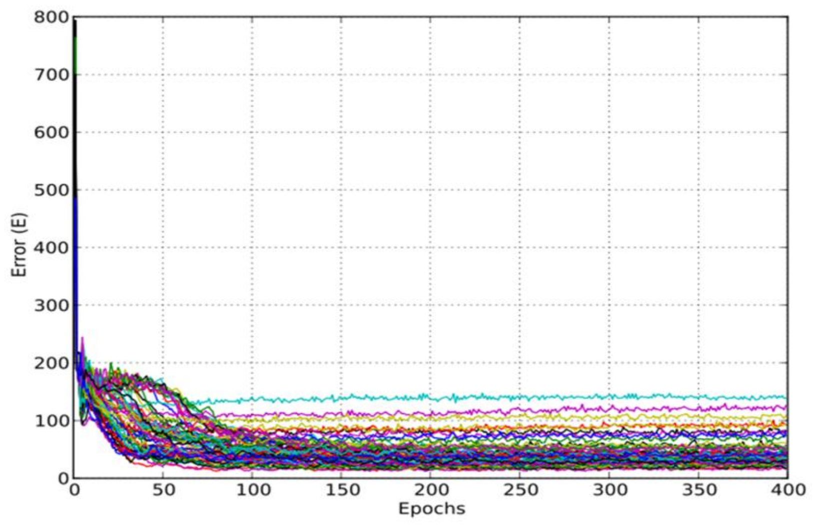
Performance is evaluated by SPAN ability to train the network so it generates the target train with the lowest percentage of error and in the fastest time (Mohemmed, Schliebs & Kasabov, 2011). As for the memory experiment, performance is evaluated by having the highest load factor (Mohemmed, Schliebs & Kasabov, 2011). These will be explained in more detail in the following results and analysis section.

4.2.3 Results and analysis:

After running the experiments, the results are compared and analysed. In the learning multiple spikes experiment, Figure 8 shows the Error E versus the number of epochs for learning the spiking input train to match the target train. Figure 8.A is the result using the static synapses, while Figure 8.B shows the result of SPAN with sixty four configurations for the dynamic synapses. However, as visually identifying the configuration for each line in Figure 8.B is difficult, they need to be divided into groups.



A.



B.

Figure 8 The error vs. the number of epochs for learning a spiking train using SPAN, Diagram A is with static synapses and Diagram B is with dynamic synapses with diffrent configration setups.

For a better visualization and analysis of the results obtained using the dynamic synapses, Figure 9 simply divide the results into four groups based on the configuration of the probability of the dynamic synapses. These groups are shown in Figure 9. Figure 9.A, Figure 9.B, Figure 9.C and Figure 9.D have the probability values of 1, 0.75, 0.5 and 0.25 respectively.

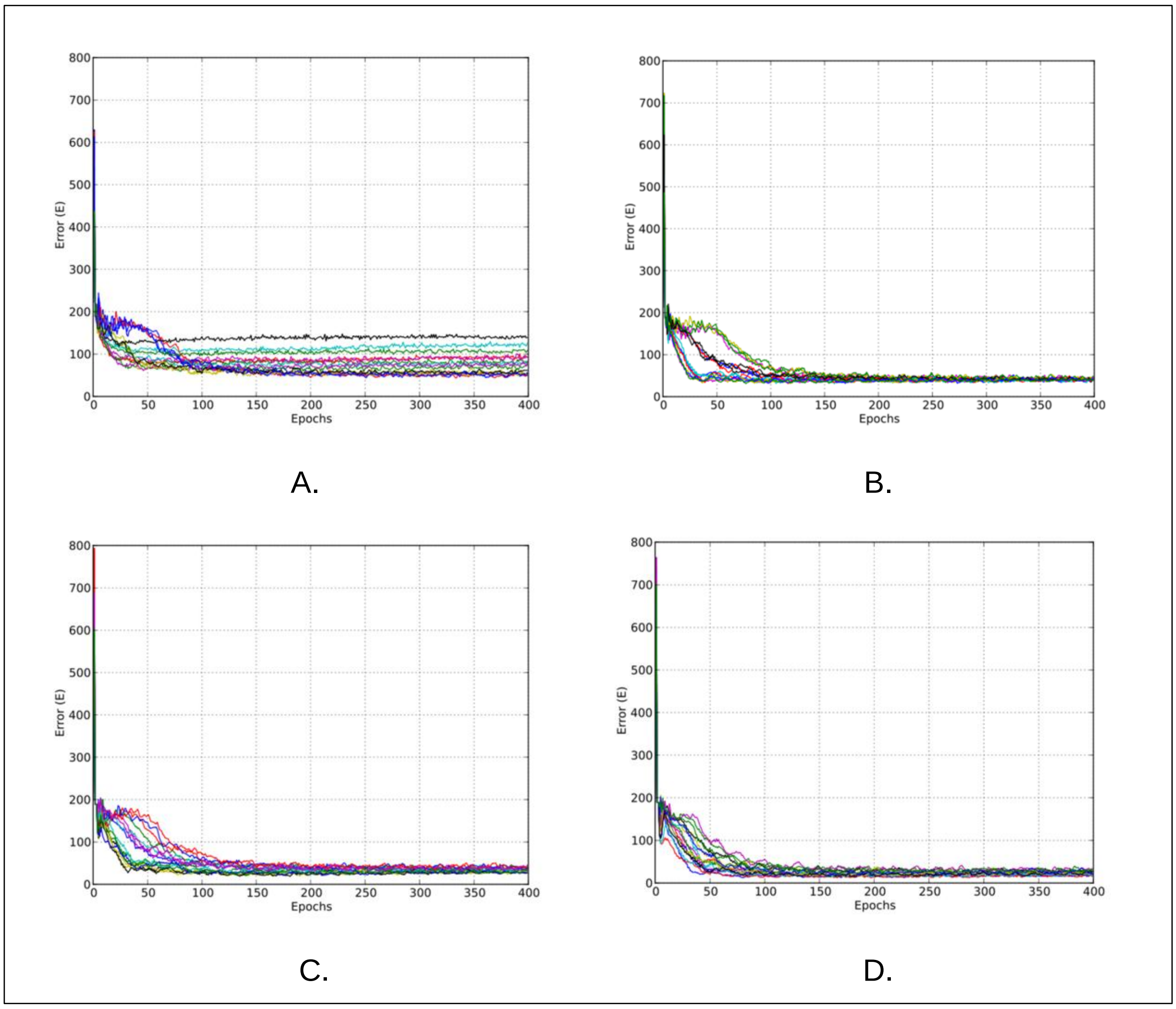
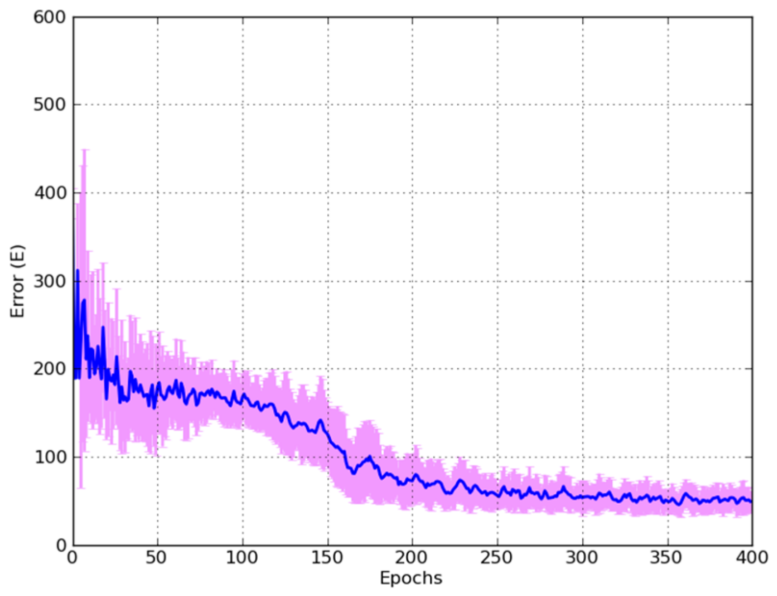
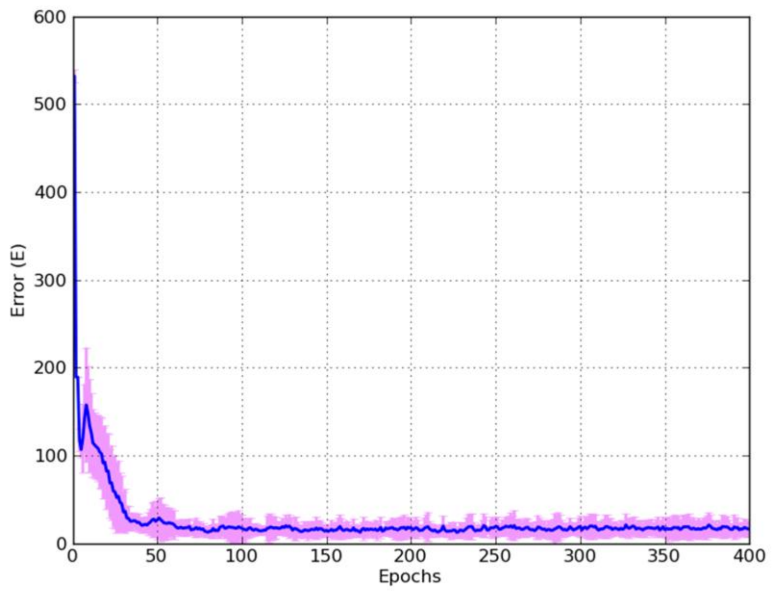


Figure 9 The SPAN result from the dynamic synapses is divided into four groups, the groups are divided according to the probability value u, (A): u = 1, (B): u = 0.75, (C): u = 0.5, (D): u = 0.25.

It can be observed from Figure 9 that the group that has stabilized with the lowest error result is group D which has the probability of u = 0.25. This result is expected due to the mathematical function used. Next, the group was observed closely and the lowest results from the group and their configurations are selected and identified for the memory experiment. The configuration with the lowest error has the values of ($u = 0.25$, $\tau_{rec} = 1200$, $\tau_{fac} = 50$). These values are selected based on the process of viewing and eliminating the highest results in the plotted diagram in Figure 9.D even though the differences between the lowest results are insignificant.



A.



B.

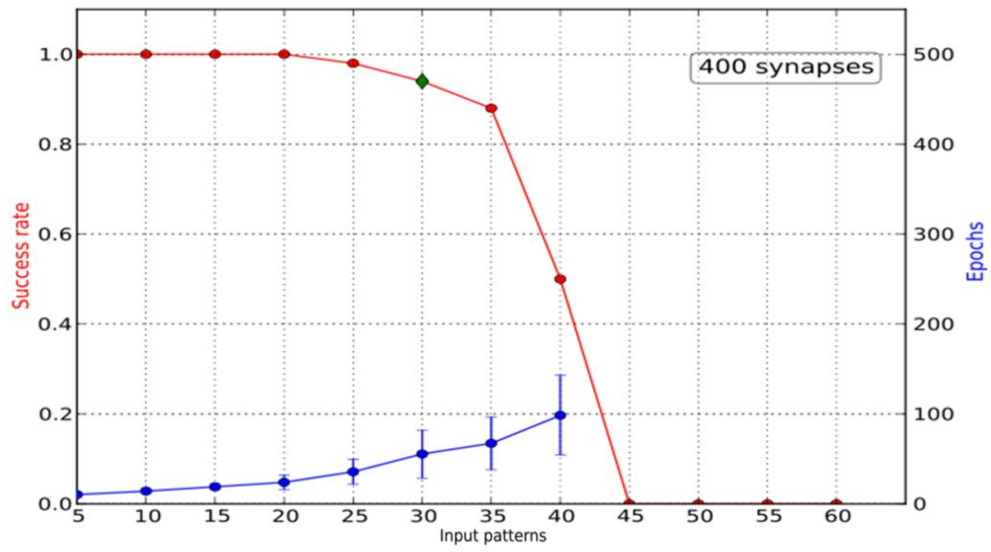
Figure 10 The average and standard deviation of the of the resulting error vs. the number of epochs. (A) is SPAN with the static synapses and (B) is SPAN with dynamic synapses.

In Figure 10, the result and its standard deviation is plotted. Figure 10.A shows the result using static synapses and Figure 10.B using dynamic synapses. Using the dynamic synapses in Figure 10.B showed that it reaches a stable result in approximately 50 epochs. On the other hand, it reaches to a stable result after 250 epochs with the static synapses. The diagram also illustrates in this case that the error rate value of SPAN with dynamic synapses is less than that with static synapses, having an error result around 18 with dynamic synapses and around 50 with static synapses. Furthermore, the experiment shows fluctuation is more obvious with static synapses and is less so when using dynamic synapses. Thus, this selected optimal setting appears to have a strong decreasing effect on the synaptic weight. In other words, the best setting is the one that decreases the synaptic efficacies in the network the most.

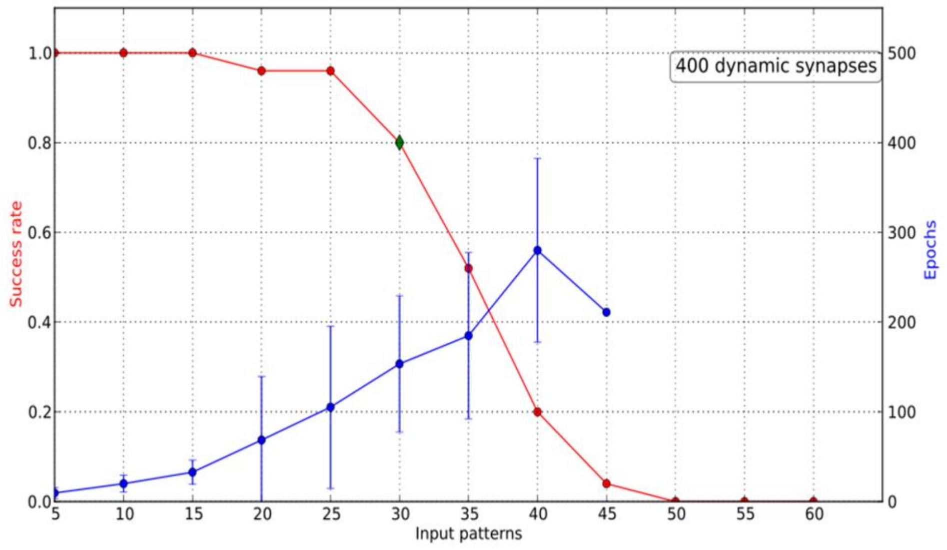
Although using dynamic synapses with SPAN indicates it could learn multiple input trains more than two times better than SPAN with static synapses, the loading factor in the memory test shows unexpectedly different results.

The results of the second experiment, the memory experiment, are shown in Figure 11. The two plots in Figure 11 illustrate the average results of 25 trails. Figure 11.A is the result using static synapses while Figure 11.B is the result using dynamic synapses. Both plots report the success rate, which are the red curves, if the input pattern is correctly classified for the number of trails. Moreover, the plots reported the average number of epochs required to learn the correctly classified inputs and are shown in blue.

The load factor $\frac{p}{n}$ is created where the success rate is 90% or above, which is indicated by the diamond marker. After calculation, the load factor is equal to 0.075 when using static synapses and is also equal to 0.075 when using dynamic synapses, suggesting there are no major differences between them. However, in Figure 11 A the result shows when using static synapses, SPAN still has a success rate of over 80% even when using 35 input patterns. When using dynamic synapses, the success rate dropped to around 50%. Meaning that, in this experiment, SPAN with static synapses is able to learn more input patterns than SPAN with the dynamic synapses.



A.



B.

Figure 11 The load factor results and the number of epochs required for the memory experiment for SPAN, diagram (A) with using static synapses and diagram (B) is with dynamic synapses.

Moreover, the plots show that SPAN with static synapses used fewer epochs for obtaining the results. On the other hand, in the case of the dynamic synapses in this experiment, the plots showed that SPAN used more epochs. For example, in Figure 11, when using 30 input patterns, SPAN used an average of around 70 epochs with the static synapses, and an average of around 160 epochs when using the dynamic synapses. In this case, it is more than double the epochs used. Therefore, SPAN with static synapses is faster in this experiment.

The results gained from the memory experiment showed that using dynamic synapses does not produce better results than using static synapses. This challenge the hypothesis that using dynamic synapses will lead to better results over static synapses. However, the load factor obtained is 0.075 for either type of synapse for this experiment, but the result is not higher than the result obtained by Chronotron done by Florian (2010) , which is 0.22. However, they are higher than the results gained with ReSuMe, which is between 0.02 and 0.04 in the paper by Florian (2010). Furthermore, using the dynamic synapses with SPAN for this experiment, required more training epochs than when using static synapses. This means that it is requiring more resources for achieving the same result than when using static synapses. This might be the possibility of initializing a lower synaptic weight value for the static synapses in the memory experiment. Table 5 summarizes the main results obtained with SPAN.

Table 5 Summury of the experiments' main results with SPAN

	Learning multiple spike train experiment		Memory experiment	
	Average epochs required to reach saturation	Last error E value at saturation	Load factor	Average epochs required to learn 30 input patterns
SPAN with Static synapses	≈ 250 epochs	≈ 51	0.075	≈70 epochs
SPAN with dynamic Synapses	≈ 50 epochs	≈ 20	0.075	≈160 epochs

Even though the configuration of the dynamic synapses was changed three times to see if the memory experiment may produce a different outcome, the output results still conclude that using the dynamic synapses with SPAN does not report a better result than using static synapses. However, these new changes to the configuration did not demonstrate any significant difference than the configuration that was reported in this paper, thus, they were not included in the report.

There are also other variables that might change the scenario of the outcome. For example, using a different target train and using different classes may have other scenario outcomes. That is because, in the learning multiple spikes experiment, the SPAN with dynamic synapses showed a better result with a longer train sequence, and it may produce a similar outcome in the memory experiment. In addition, since the chosen optimal settings for the dynamic synapses decreased the synaptic weight, it is argued that by lowering the synaptic weight of the static synapses could results lower and better E values in SPAN in the first experiment. However, we repeated the first experiment with two lower weights, we used weight 1 and 2.5 and there was no improvement . Then agin, additional experiments are needed.

A possible issue with SPAN is that the dynamic synapses parameters were not optimized. The new hypothesis is, when using a method similar to PSO for optimizing the dynamic synapses parameters, the load factor might get improved in the memory experiment, and the optimization might also decrease the speed performance in learning multiple spike patterns, because of the additional step required. However, in Cronotron approach, speed performance was acceptable even though they optimized the dynamic synapses parameters and the synaptic weight. Still, this can only be confirmed by a further investigation and experiments.

CHAPTER 5

5. Conclusion and future directions:

5.1 Conclusion:

In conclusion, the main objective has been achieved. The report started with a description of SNN and its components such as integrate and fire neuron models, the dynamic synapses models, the binless fitness functions. Their advantages and possible disadvantages and limitations were identified. Moreover, the different supervised learning algorithm approaches that can be used with SNN such as PSO and SPAN were explained. After that, the experiments setup were described and the parameters tuning were defined. Then, comparisons of the results gained when training SNN with SPAN were presented and analysed.

The results presented in this paper suggest that Spiking Neural Networks with dynamic synapses can successfully learn spatial temporal data and are able to memorize and classify spike trains as observed with SPAN, and were able to perform better than other algorithms, i.e. ReSuMe in the memory capacity result, but was far from the results obtained by Chronotron. Working with PSO showed that it is difficult to tune the dynamic synapses' parameters even though it successfully trained the input train.

Furthermore, considering the set of data used in the experiments, the results suggest that combining more biological plausible component like dynamic synapses with SPAN, it did not always achieve the best results, such as in the memory experiment, implying that it is better to work with static synapses and to keep tuning the weight per synapse. On the other hand, if memory capacity is not needed, then the results suggest that SPAN is faster and have lower E values when using dynamic synapsis, but the right parameters values need to be set correctly. However, this one example is not enough and more experiments are needed with different data sets to confirm this conclusion.

5.2 Future directions:

Even though PSO is inspired by natural behaviour, SPAN proved more interesting in this study. It is a much simpler model than the model using PSO. Furthermore, SPAN showed that it can classify sequence trains more successfully than other algorithms like ReSuMe, and experimenting further with SPAN in the future to improve the algorithm with dynamic synapses by tuning the neuron parameters and the synaptic weights to obtain the best result is an interesting challenge. Applying SPAN with dynamic synapses to solve real-world datasets or to classify temporal data similar to the problem solved with liquid state machines algorithms is an exciting task to work on, such as working with video data or voice recognition data. Moreover, encoding these data and learning encoding schemes for discovering new knowledge will add to the learning experience and will be worth taking into consideration. In addition, a possible future direction worth exploring is researching other machine learning algorithms and combining them with SNN such as, evolving classification and fuzzy learning, which initially showed impressive results with data classification in my other previous research paper. Investigating the possibilities for implementing a hybrid model with SPAN to create a robust method and achieve efficient results and better performance is inspiring. Neuromorphic engineering is a new research field that might be a possible future research area. Building spiking neuron networks and learning models in a very large scale integration (VLSI) would open up more challenges and possibilities worth consideration.

References:

Belatreche, A., Maguire, L. P., & McGinnity, M. (2006). Advances in Design and Application of Spiking Neural Networks. *Soft Computing*, *11*(3), 239–248.

Bohte, S., & Kok, J. (2005). Applications of spiking neural networks. *Information Processing Letters*, *95*(6), 519–520.

Bohte, S.M., Kok, J. N., & La Poutré, H. (2000). SpikeProp: backpropagation for networks of spiking neurons. *Proceedings of the Twelfth Belgium-Netherlands Artificial Intelligence Conference (BNAIC)* (p. 321).

Bohte, Sander M. (2004). The Evidence for Neural Information Processing with Precise Spike-times: A Survey. *NATURAL COMPUTING*, *3*, 2004.

Carlisle, A., & Dozier, G. (2001). An off-the-shelf PSO. *Proceedings of the workshop on particle swarm optimization* (Vol. 1, pp. 1–6).

Carvalho, M., & Ludermir, T. B. (2006). Particle Swarm Optimization of Feed-Forward Neural Networks with Weight Decay (p. 5). Presented at the Sixth International Conference on Hybrid Intelligent Systems, 2006. HIS ’06, IEEE.

Chen, G., & Yu, J. (2005). Particle Swarm Optimization Neural Network and Its Application in Soft-Sensing Modeling. In L. Wang, K. Chen, & Y. S. Ong (Eds.), *Advances in Natural Computation* (Vol. 3611, pp. 610-617). Berlin, Heidelberg: Springer.

Dauwels, J., Vialatte, F., Weber, T., & Cichocki, A. (2009). On Similarity Measures for Spike Trains. In M. Köppen, N. Kasabov, & G. Coghill (Eds.), *Advances in Neuro-Information Processing* (Vol. 5506, pp. 177–185). Berlin, Heidelberg: Springer.

Florian, R. V. (2010). The chronotron: a neuron that learns to fire temporally-precise spike patterns. Retrieved from http://precedings.nature.com/documents/5190/version/1/files/npre20105190-1.pdf

Gerstner, W., & Kistler, W. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity.* Cambridge University Press.

Gewaltig, M.-O., & Diesmann, M. (2007). NEST (NEural Simulation Tool). *Scholarpedia*, *2*(4), 1430. doi:10.4249/scholarpedia.1430

Goodman, E., & Ventura, D. (2006). Spatiotemporal Pattern Recognition via Liquid State Machines. *International Joint Conference on Neural Networks, 2006. IJCNN '06* (pp. 3848–3853). Presented at the International Joint Conference on Neural Networks, 2006. IJCNN '06, IEEE.

Gütig, R., & Sompolinsky, H. (2006). The tempotron: a neuron that learns spike timing-based decisions. *Nature Neuroscience*, *9*(3), 420–428.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*(6), 1569– 1572.

Jim-Shih Liaw, & Berger, T. W. (1997). Computing with dynamic synapses: a case study of speech recognition. *Neural Networks,1997., International Conference on* (Vol. 1, pp. 350–355 vol.1). Presented at International Conference on the Neural Networks,1997. IEEE.

Jin, Y., Wen, R., & Sendhoff, B. (2007). Evolutionary multi-objective optimization of spiking neural networks. *Proceedings of the 17th international conference on Artificial neural networks*, ICANN'07 (pp. 370–379). Berlin, Heidelberg: Springer.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Neural Networks, 1995. Proceedings., IEEE International Conference on* (Vol. 4, pp. 1942–1948 vol.4). Presented at IEEE International Conference on Neural Networks, 1995. Proceedings.

Mohammed, A., Schliebs, S., & Kasabov, N. (2011). SPAN: A Neuron for Precise-Time Spike Pattern Association. In B. -L. Lu, L. Zhang, & J. Kwok (Eds.), Neural Information Processing (Vol. 7063, pp. 718-725). Berlin, Heidelberg: Springer.

Mohemmed, A., Matsuda, S., Dhoble, K., & Kasabov, N. (2011). Optimization of Spiking Neural Network with Dynamic Synapses for Spike Sequence Generation using PSO. *The 2011 International Joint Conference on Neural Networks (IJCNN)* (pp.2969-2974). Presented at the 2011 international Joint Conference on Neural Networks (IJCNN), IEEE.

Mohemmed, A., Schliebs, S., Matsuda, S., & Kasabov, N. (2011). Method for Training a Spiking Neuron to Associate Input-Output Spike Trains. In L. Iliadis & C. Jayne (Eds.), *Engineering Applications of Neural Networks* (Vol. 363, pp. 219–228). Berlin, Heidelberg: Springer.

Natschläger, T., & Maass, W. (2002). Spiking neurons and the induction of finite state machines. *Theoretical Computer Science*, *287*(1), 251–265.

Paiva, A. R. C., Park, I., & Príncipe, J. C. (2009). A comparison of binless spike train measures. *Neural Computing and Applications*, *19*(3), 405–419.

Paugam-Moisy, H. (2006). Spiking neuron networks: a survey. *Rapport Technique RR-11, IDIAP,* Martigny, Switzerland.

Paugam-Moisy, H., & Bohte, S. M. (2009). Computing with spiking neuron networks. *Handbook of Natural Computing, 40p.* Heidelberg: Springer.

Pavlidis, N. G., Tasoulis, O. K., Plagianakos, V. P., Nikiforidis, G., & Vrahatis, M. N. (2005). Spiking neural network training using evolutionary algorithms. *International Joint Conference on Neural Networks, 2005.*

IJCNN ’05 (Vol. 4, pp. 2190–2194 vol. 4). Presented at the International Joint Conference on Neural Networks, 2005. IJCNN ’05. Montreal, Canada.

Ponulak, F., & Kasinski, A. (2006). ReSuMe learning method for Spiking Neural Networks dedicated to neuroprostheses control. *In Proc. of EPFL LATSIS Symposium 2006, Dynamical principles for neuroscience and intelligent biomimetic devices,* (pp. 119–120).

Ponulak, F., & Kasiński, A. (2010). Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting. *Neural computation*, *22*(2), 467–510.

Sichtig, H. (2007). Building Smart Machines by Utilizing Spiking Neural Networks; Current Perspectives. *Computational Intelligence and Bioinformatics and Computational Biology, 2007. CIBCB ’07. IEEE Symposium on* (pp. 346–350). Presented at the Symposium on Computational Intelligence and Bioinformatics and Computational Biology, 2007. CIBCB ’07. IEEE.

Sun, R., & Giles, C. L. (2001). Sequence learning: from recognition and prediction to sequential decision making. *Intelligent Systems, IEEE*, *16*(4), 67–70.

Tsodyks, M., Pawelzik, K., & Markram, H. (1998). Neural Networks with Dynamic Synapses. *Neural Computation*, *10*(4), 821–835.

van Rossum, M. C. W. (2001). A Novel Spike Distance. *Neural Computation*, *13*(4), 751–763.