
Pulsed Neural Networks and their Application

Daniel R. Kunkle

drk4633@rit.edu
<http://www.rit.edu/~drk4633/>

Chadd Merrigan

ccm2451@cs.rit.edu
<http://www.rit.edu/~ccm2451/>

Computer Science Dept.
College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY, 14623

February, 2002

Abstract

Pulsed neural networks are networks of spiking neurons, which represent an entirely new class of artificial neurons. Here we present an overview of pulsed neural networks, including the structure, function and available training mechanisms for networks of spiking neurons. We highlight differences between this model, “first generation” threshold gates, and “second generation” sigmoid activation gates, and we examine current research into pulsed neural networks and the use of temporal information in neural processing. Lastly, we summarize our own research toward the end of using pulsed neural networks to identify computer users by the cadence of their keystrokes.

1 INTRODUCTION

Artificial neural networks, being based on the workings of biological neural networks, can be expected to draw inspiration from advances in neurophysiology and related fields. Traditionally, it was believed that neuron communicated information in their mean firing rate. In other words, one neuron would receive information from another by “counting” the number of spikes from that neuron over some extended period of time and determining the mean time between firings. Specifically, a shorter time period implies a higher activation. This type of information coding is known as *rate coding* and has been one of the dominant tools for measuring neuron activity over the past 80 years of neurological study (Gerstner, 1999). Correspondingly, a majority of artificial neural network models have used rate coding, in the form of real number values representing an activation level.

Recently, new discoveries and advances in neurophysiology have encouraged new explorations in alternative information coding schemes in artificial neural

networks. Neurons have been found in the primate brain that respond selectively to complex visual stimuli after as few as 100-150 ms after the stimulus was presented. This information must have passed through approximately 10 layers of processing between the initial photoreceptors and the neurons that selectively respond to the stimuli. Given this, it was argued that each individual processing stage would have only 10 ms to complete and that this amount of time is insufficient for rate coding as a means of information passing (Thorpe et al., 2001).

Another information coding scheme must therefore be at work in the brain, one that may be useful in artificial neural networks. Codes based the temporal relationship between the firing of neurons is a promising alternative. Using such codes it is possible to transmit a large amount of data with only a few spikes, as few as one or zero for each neuron involved in the specific processing task (Thorpe et al., 2001). It is the use and effects of temporal information coding in artificial networks that will be examined here.

Section 2 comprises an overview of artificial spiking neurons, including the main categories of artificial neurons, a brief overview of the biological basis of the spiking neuron model and the main benefits and qualities of using artificial spiking neurons.

In section 3 the model of an artificial spiking neuron is examined in depth, detailing its structure and functioning, and introducing the common mathematical model.

Section 4 presents methods for training spiking neurons and pulsed neural networks, allowing them to learn and act on temporally encoded data.

Finally, sections 5 and 6 explore possible applications of pulsed neural nets, including some original work by the authors aimed at identifying computer users by the cadence of their keystrokes.

2 OVERVIEW OF SPIKING NEURONS

2.1 Artificial Neuron Generations

Wolfgang Maass (Maass, 1997) delineates past and current artificial neural network research into three generations and makes the following observations.

The first generation is based on the McCulloch-Pitts neuron (also known as a perceptron or a threshold-gate) as the basic computation unit. Models of the first generation, such as the multi-layer perceptron, use digital input and output, usually binary or bipolar. Any boolean function can be computed by some multi-layer perceptron with a single hidden layer.

The second generation is based on computation units (neurons) that use an activation function of a continuous set of possible output values. Commonly, these activation functions are the sigmoid, $f(x) = 1 / (1 + e^{-\sigma x})$, or the hyperbolic tangent, $f(x) = (1 - e^{-2x}) / (1 + e^{-2x})$. Second generation neural networks, like first generation networks, can compute arbitrary boolean functions (after using a threshold). Second generation networks can compute certain boolean functions with fewer neurons than first generation neurons. Also, second generation networks with one hidden layer can approximate any continuous, analog function arbitrarily well. Important to many implementations is the fact that second generation networks support learning algorithms based on gradient descent, such as error back-propagation.

The third generation of artificial neural networks is based on spiking neurons, or “integrate and fire” neurons. These neurons use recent insights from neurophysiology, specifically the use of temporal coding to pass information between neurons. These networks, like those of the second generation, can approximate continuous functions arbitrarily well, but with temporally encoded inputs and outputs (Maass, 1997; Maass, 1999). Further, there are functions that require fewer neurons in a pulsed neural net to approximate than would be needed in a second generation network (Maass, 1997).

All three of these generations are simplifications of what is known about the physiology of biological neurons but the third generation is the model with the highest fidelity.

2.2 Biological Basis

As was mentioned previously, one of the main impetuses for considering the use of temporal codes is the speed with which they can transmit data when compared to rate coding. Many tasks performed by biological neural networks are completed in a very short time, often a period so short that only one firing per neuron could be sampled. This is demonstrated by figure 1. Note the low number of spikes from any individual neuron over the four second period.

Besides the possible speed improvements, biological systems have other reasons to use temporal codes. Very

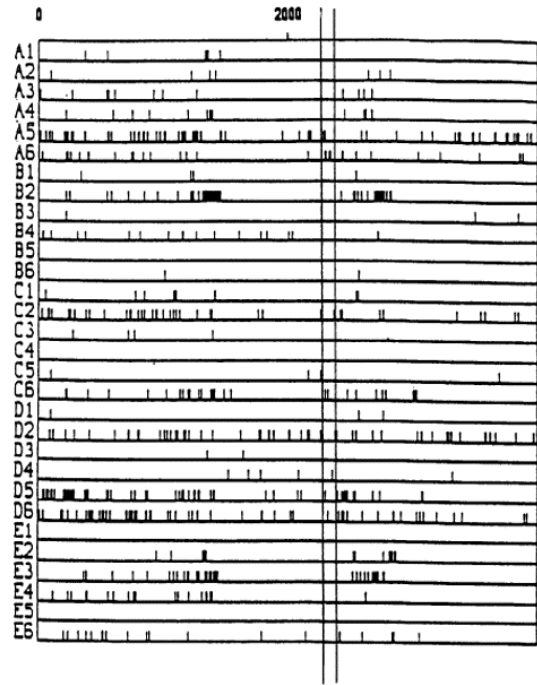


Figure 1: Simultaneous recordings (over 4 seconds) of the firing times of 30 neurons from monkey striate cortex (Krüger and Aiple, 1988). Each firing denoted by a short vertical bar, with a separate row for each neuron. For comparison, two vertical lines mark a length of an interval of 100 msec. The time span is known to suffice for the completion of some complex multilayer cortical computations (reproduced from Maass, 1997).

often there is information inherent in the differences in timing of stimuli perceived by a biological system. In (Thorpe et al., 2001) a number of natural situations where this stimuli time differential are presented, including sound localization where differences in the arrival times of stimuli to the left and right ear provide a means to locate the source, and motion perception in the visual system by means of timing differences between different photoreceptor responses.

Many biological neural systems not only use temporal information but do so with incredible precision and accuracy. For example, the auditory system of the barn owl has the capability to locate sources of sound in the horizontal plane with a precision of 1 to 2 degrees. This equates to a temporal difference of only a few microseconds ($< 5\mu s$) between the arrival of sound waves at the left and right ears (Gerstner et al., 1999).

This ability to learn and act in a dynamic environment, rich with temporal information, is a necessary quality for biological systems and for artificial systems that seek to perform similar tasks.

2.3 Benefits of Pulsed Neural Networks

Speed. As has been presented, networks composed of spiking neurons can transmit and receive substantial

amounts of data through the relative timing of only a few spikes. This leads to the possibility of very fast and efficient implementations.

Real-Time Action. Pulsed networks are designed to use temporal information and so can integrate readily into “real” dynamic environments.

Complexity. Pulsed networks can compute any function a second generation network can and can often do so with fewer neurons.

Biological Fidelity. Because pulsed networks adhere more closely to what is already known about biological neural networks, they can benefit more readily from the rapidly increasing base of knowledge gained from the field of neurophysiology.

3 SPIKING NEURON MODEL

Artificial spiking neurons are a fairly new technology, being used extensively in only the last five to ten years. Most of the publications regarding spiking neurons still focus on research rather than implementation. Therefore a standardized model has not coalesced. Such a standard may never emerge, because the category of spiking neurons is already broad enough to include several important subcategories. Here we describe the most salient features common to all models and introduce some of the more important variations.

3.1 Behavior

Let us first look at the behavior of an individual neuron over time. An artificial spiking neuron has a value analogous to the membrane electrical potential in a real neuron, which is typically referred to as its potential or $P_v(t)$ to indicate a potential for neuron v at a certain time t . This potential is equal to the sum of its excitatory postsynaptic potentials (referred to as EPSPs) and its inhibitory postsynaptic potentials (IPSPs). The inhibitory potentials being negative, they tend to lower the resultant potential. If this potential becomes large enough, it may pass a given threshold potential, at which point the neuron will fire (fig. 2); thus they are often referred to as *integrate and fire* neurons.

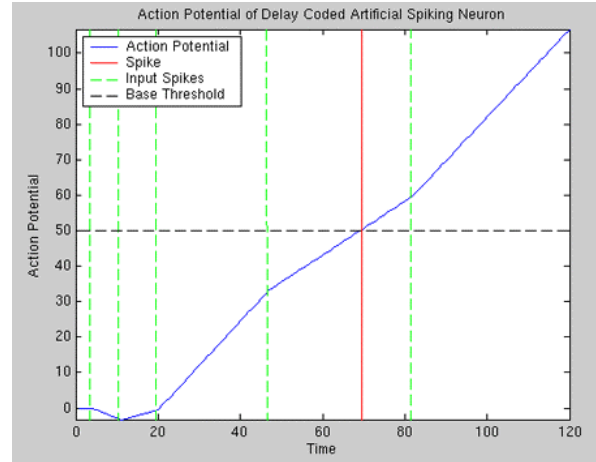


Figure 2: Potential rising past a threshold and firing. The neuron will not fire again until the refractory period, determined by τ_v , has ended.

3.2 Threshold functions

As mentioned earlier, natural neurons do not usually exceed a firing rate of around 100 Hz, so clearly the neuron does not fire at all possible times when its potential is above the resting threshold. In nature as well as in the spiking model, a refractory period following each firing prevents the neuron from firing for a minimum duration, apparently around 10 msec in a real neuron. In the spiking model, this phenomenon is mimicked by the use of the τ_v function which defines the effective threshold potential of the neuron. As shown in figure 3, the value of the effective threshold shoots up immediately to infinity upon firing, thereby preventing any potential from being above the effective threshold. After a minimum duration, called the *absolute refractory period*, the τ_v function defines a period where the neuron remains reluctant to fire, the *relative refractory period*. Figure 3 shows this function as it would be found in nature, with a slope delineating the relative refractory period. For the purposes of much research, however, the approximate model on the right will suffice. This can simplify calculations for proof and simulation when the neurons can be assumed to fire far apart in time, or in other words, the value of $(t - t')$ at the next firing time will be sufficiently large that $\tau_v(t - t')$ will effectively be 0 in all cases.

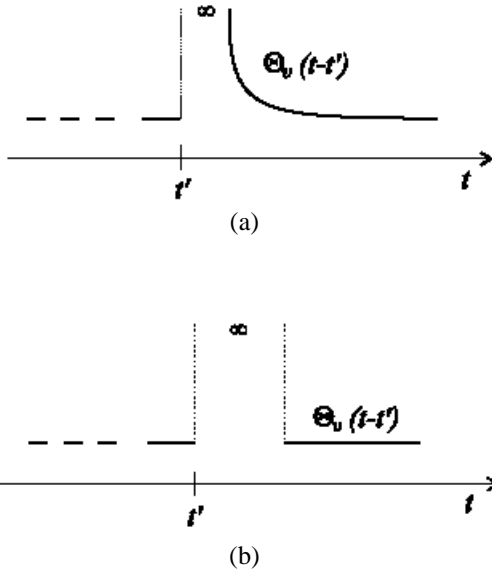


Figure 3: (a) Typical shape of a biological threshold function. (b) Effective approximation of this function (Maass, 1997).

3.3 Response Functions

The voltage of one spike arriving at an EPSP over time is modelled below. It has a distinctly non-vertical leading edge and a less steep descent thereafter, approaching 0 volts more and more gradually.

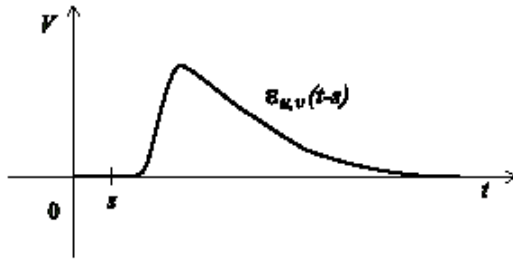


Figure 4: Shape of a biological response function (Maass, 1997).

One spike is generally not enough to cause a postsynaptic neuron to fire. Biological neurons typically must have their potentials raised by around 20 mV, while individual spikes change this potential by a few millivolts at most (Maass, 1997). Therefore to determine $P_v(t)$ it is necessary to integrate the response function for each spike encountered before time t . This could be a daunting task as spikes accumulate over a simulation. Accordingly, it is

customary to define a length of time after which the response function can be approximated as zero.

Common response functions include a step, or *piecewise constant*, function, shown as type A in dark blue on figure 5 below; a *continuous and piecewise linear* function, type B in green below; and type C, any *continuously decreasing* function such as this one $e^{\frac{-4t}{t-s}}$ proposed in (Jahnke et al., 1999), shown in red.

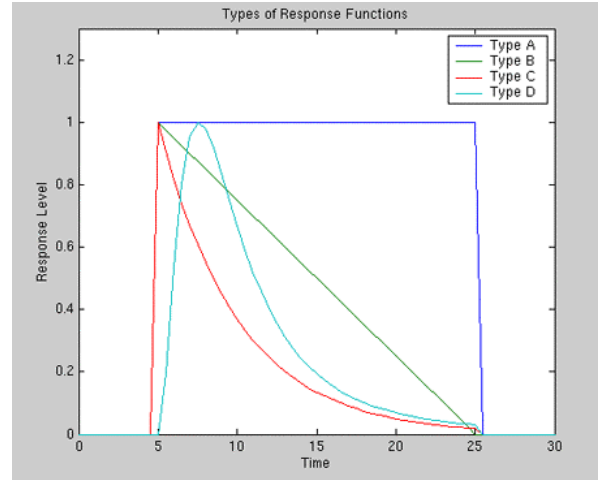


Figure 5. Response functions of increasing fidelity to natural responses. Time 0 is equivalent to s ; the point where each graph ascends is point s' .

Each level of fidelity to the natural function shown above adds more functionality to the network. Maass has proven that some functions impossible to approximate with type A neurons can be handled by type B (Maass, 1997). Since the pattern here and in the advances of artificial neural networks in general implies that fidelity to nature improves capability, we propose the following more refined model which more closely mimicks the natural response function, shown in blue above:

$$\frac{-\cos\left(\frac{\frac{6+4\sqrt{2}}{\pi}}{\left(\left(\frac{t-s'}{t_{end}-t'}\right)^2 + \frac{1+\sqrt{2}}{\pi}\right)}\right)+1}{2}$$

where $t - s'$ represents the time since the spike reached the neuron, $t_{end} - t'$ represents the duration of the applicable response function values. This equation effectively stretches a cosine wave shrunk to $[0, 1]$ onto the range of

all positive numbers $(t - s')$ using a quadratic mapping function of the form $\frac{\beta}{(x + \alpha)^2}$.

3.4 Weights

As in the models of the previous two generations of artificial neurons, the input potential $P_v(t)$ a spiking neuron v are mediated by a weight w_{ij} where i is the index of the presynaptic neuron for which this w applies. Generally, this is used as a multiplicand with the value of $\varepsilon(t - s)$ to arrive at the effective EPSP or IPSP for the connection from u_i to v_j as it is in the perceptron model. However, it has been shown that real synapses do not fire in a deterministic manner (Maass, 1997). Rather, the property represented by our w value would be analogous to a probability of firing at a real synapse. The vesicles containing neurotransmitters to cross the synaptic gap release their contents completely, yet the release is not guaranteed to happen consistently. There are currently many researchers working with these stochastic networks which are essentially the same as networks known as the *spike response model*. For our relatively primitive experiments, we treated the weights as deterministic analog values.

3.5 Excitatory vs. Inhibitory Neurons

In addition to the weight factor, artificial spiking neurons can have an excitatory or inhibitory effect on the potential of their output connections. In the inhibitory case, $\varepsilon(t - s)$ is simply a negative image of the excitatory version. Recent research has indicated that neurons in the brain normally have one or the other effect on each neuron which receive their outputs. Therefore, this variable can be modeled as a value k_u which is equal to 1 or -1 for a particular neuron u . In practice, this k value is simply assumed to be part of the $\varepsilon_{u,v}(t - s)$ response function at the synapse $\langle u, v \rangle$.

3.6 Formal Specification for $P_v(t)$.

Reviewing the variables described above, we can compose the formal definition of the potential value $P_v(t)$.

(a) The effect of one spike traveling from u to v :

$$\varepsilon_{u,v}(t - s)$$

(b) The effect of several spikes arriving at one synapse $\langle u, v \rangle$

$$\sum_{n: s_n \in F_u} \varepsilon_{u,v}(t - s'_n)$$

where F_u is the set of firing times of neuron u .

(c) The effect of spike trains from several input neurons $u_1 \dots u_n$:

$$\sum_{i: u_i \in \Gamma_v} \sum_{n: s_n \in F_{u_i}} \varepsilon_{u,v}(t - s'_n)$$

where Γ_v is the set of all input neurons to v .

(d) Finally, with the effect of weights w_{ij} :

$$P_v(t) = \sum_{i: u_i \in \Gamma_v} \sum_{n: s_n \in F_{u_i}} \varepsilon_{u,v}(t - s'_n) w_{ij}$$

3.7 Information Coding

Clearly the spiking model is fundamentally different than previous generations of artificial neurons. Most importantly, the information passed by spikes can only be that of the relative timing between them. Thus the passing of useful information across a spiking net requires conversion from other forms (typically analog) to temporal data. We will see that some methods of coding rely on relative timing alone, but that others rely on a synchronized, known time and input interval. While some of the absolute time dependent models show encouraging results, real neurons do not have this facility and we can assume that an effective spiking network need not include such absolute time sensitivity. On the other hand, it has been shown that synchronization of spike trains can be accomplished in large populations of neurons and can facilitate learning of in some networks (Maass, 1999).

The most familiar and intuitive form of coding is known as *delay coding*. Delay coding arises from the observation that in the natural world, more highly stimulated neurons tend to spike *sooner*, in addition to more frequently. This model relies on an absolute time frame, beginning at T_{in} and lasting for a period c . From whatever form the input data arrived, a standard sigmoid or hyperbolic tangent function must map the input onto the range $[0,1]$. After this transformation is made, the input x_j is converted to a time $T_{in} - cx_j$ when the neuron will fire.

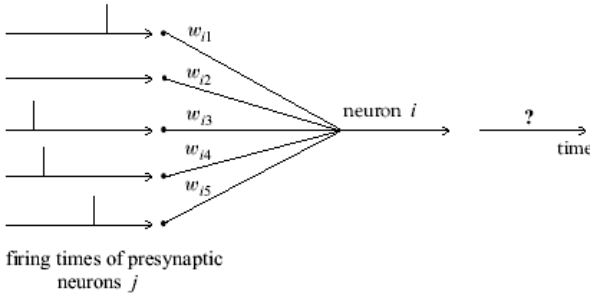


Figure 6: A delay-coded neuron (Maass, 1997).

c represents the time interval for valid input and output spikes. Input spikes outside this time interval are not considered. Generally, the earlier the presynaptic neurons fire, the earlier the postsynaptic one will fire. Thus the first firing of the neuron is determined by the timing and synergistic coincidence of the input spikes. While it is possible that the output neuron will spike more than once, its output is determined by the first. If that time t_j is

applied to the function $y_j = \frac{T_{out} - t_j}{c}$, y_j will be a

value in the range $[0,1]$ which can then be reconverted to the proper output data range.

The *binary coding* scheme is also based on knowledge of an absolute time value. In that model, spikes fill out bits in a numeric value. As small time segments pass, those segments with a spike occurring are considered to represent 1; segments with no spikes represent 0.

A *count code*, where values are represented by the number of spikes within a given time interval, is also absolute time dependent.

The last absolute time dependent encoding method is called the *timing code*. This depends on being able to determine very accurately the exact time of fire, such that each spike could convey a value with information equivalent to the inverse of the granularity of the distinguishable time intervals. We did not experiment with these models, as our interests are in the paradigms more useful for real-time response.

Simon Thorpe and his team (Arnaud Delorme, Rufin VanRullen, Marie Fabre-Thorpe) in Toulouse, France, are at the leading edge of research into spiking networks. Among many other things, they have inspired the first commercial product, SpikeNET, a visual-recognition system. They argue that a system of *rank order coding*, where first spikes received by a neuron are given influence and later ones are inhibited, can deliver an extraordinary amount of information (Thorpe, 2000). It is immediately apparent that, as they claim, 10 input neurons with even just one spike per neuron can arrive in

10! orders. Furthermore, the model of neuronal dynamics which can accomplish a sensitivity to order is alluringly simple.

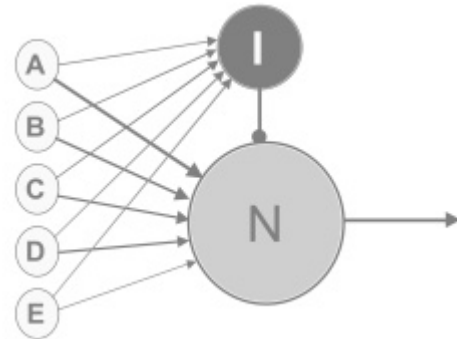


Figure 7: A simple network designed to favor the spike of the winning neuron (Thorpe, 2000).

In this model, neuron I inhibits neuron N. Therefore an accumulation of spikes which activates N will at the same time tend to activate I, whose inhibitory factor will decrease the likelihood that later spikes will cause N to fire.

Other significant research into proving qualities of various classes of spiking networks and exploring learning behaviors is being done at the university in Graz, Austria, by Wolfgang Maass, Berthold Ruf, and Heinrich Schmitt. The field is still very young but there are already many promising avenues to explore.

4 LEARNING

4.1 Unsupervised Learning

Much of the research into learning algorithms for pulsed neural networks has been focused on unsupervised learning. The aim of these algorithms is to allow the network to self-organize and eventually learn to discriminate between input patterns with no explicit identification of those patterns. This is precisely how biological networks learn. There is no outside force instructing the brain as to how it should react to each input that is presented to it. Rather, through interactions internal to the brain, the outputs of the brain become appropriate for the given inputs, it learns.

Perhaps the most studied and developed is the Kohonen Self-Organizing Map (SOM) (Kohonen, 2001). Kohonen proposes that the “feature map” created by the SOM algorithm can be used for preprocessing patterns for recognition or to project and visualize high-dimensional signal spaces on a two-dimensional display. Further, Kohonen states, “As a theoretical scheme the adaptive

SOM processes, in a general way, may explain the organizations found in various brain structures”.

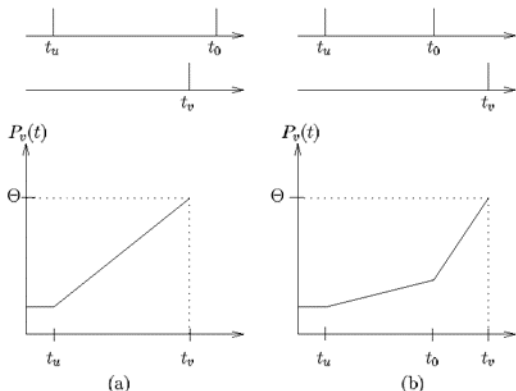


Figure 8: Spikes at synapse $\langle u, v \rangle$ and potential of neuron v during a learning cycle. In (a) the first spike of u is sufficient to make v fire before the second spike of u ; in (b) the firing of v is determined by two spikes of u . (reproduced from Ruf and Schmitt, 1997)

Very basically, the SOM is an array of nodes, each node having a *neighborhood* of other nodes. In the learning process “those nodes that are *topographically close in the array up to a certain geometric distance* will activate each other to learn something from the same input x . This will result in a local *relaxation or smoothing effect* on the weight vectors of neurons in this neighborhood, which in continued learning leads to *global ordering*” (Kohonen, 2001).

Self-organizing maps are traditionally applied in neural networks of the second generation. That is, the information is encoded as real values representing an average rate of firing of a neuron. Recently, self-organizing maps, similar to those developed by Kohonen, have been implemented using networks of spiking neurons. Ruf and Schmitt propose a such a mechanism for unsupervised learning in networks of spiking neurons (Ruf and Schmitt, 1998). This model is a further step toward biological reality in that it combines the spiking neuron model with biologically plausible self-organization. This implementation benefits from those qualities of spiking neurons presented in section 2.3, most specifically speed of computation. In the self-organizing map, a group of neurons compete with the winning neuron inhibiting the response of the other competing neurons. With a spiking neuron the winner can be computed quickly and locally, based only on single firing events.

4.2 Supervised Learning

Supervised learning has a long and very well studied history in artificial neural networks. Early training

methods for single layer networks include *Hebb learning*, *perceptron learning*, and the *delta rule*.

Hebb learning increases weights between neurons when both neurons are “on” at the same. This type of learning is one of the most simple and is originally based on biology, where this type of weight strengthening is thought to play a major role in learning. An extended version of the Hebb rule also strengthens weights when both neurons are “off” at the same time. This addition makes the algorithm more computationally powerful. The Hebb learning algorithm is applied once to each training vector and then halts. Training any more than this would have no effect.

The perceptron learning algorithm is more powerful than Hebb learning. It has been proven to converge to the correct weights if they exists for the given network configuration and problem domain. This algorithm updates the weights only when the network makes an error in classification. In the case of an error the weights are changed in proportion to a learning rate to correspond more closely to the correct answer. Perceptron learning is repeated on training vectors until no weights change.

The delta rule seeks to minimize the difference (error) between the real value output of the network and the target value. This differs from the perceptron learning algorithm, which placed a threshold on the output to determine if the network responded correctly or not.

Training using the delta rule is conducted until the error in the output is less than some threshold value.

For multi-layer networks the most successful and studied learning algorithm, by far, is error back-propagation. This algorithm is applied to feed-forward networks of neurons which have non-linear activation functions. The error between the output of the neurons and the target values is propagated backward through the network to adjust the weights of earlier connections. This method of learning is a form of gradient descent, seeking the lowest point in the error landscape.

Because Hebbian learning is biologically based, it is a natural candidate for application to pulsed neural networks. Ruf and Schmitt have presented such an application in a model of learning based on the differences in timing of the spikes of two neurons (Ruf and Schmitt, 1997). By looking at the time difference between two spikes, just one spike from both the presynaptic and postsynaptic, the benefits of using spiking neurons can be maintained. Normally, learning would occur only after an average rate of firing was determined, as in the learning methods described above for the first two generations of neural networks.

The Hebbian-type learning rule proposed by Ruf and Schmitt to change the weight $w_{u,v}$ from a presynaptic neuron u and postsynaptic neuron v is

$$\Delta w_{u,v} = \eta(t_v - t_0)$$

where t_v is the firing time of the postsynaptic neuron and t_0 is the second firing time of the presynaptic neuron (t_u is the first firing time of neuron u), and η is the learning rate. If the second firing of u , t_0 , occurs before the firing of v ($t_v - t_0 > 0$) then the weight between them, $w_{u,v}$, should be increased. If the opposite occurs, t_v is before t_0 , the weight should be decreased. This is shown by figure 8, which compares these two cases.

The assumption is that the first firing of the presynaptic neuron, u , is intended to make the postsynaptic neuron, v , fire at time t_0 , along with the second firing of u . So, the firings of the presynaptic and postsynaptic neurons are to be synchronized and the network will have learned to output the desired spike train when given a specific input.

The above method includes only one presynaptic and one postsynaptic neuron. In most situations, however, a postsynaptic neuron will have many presynaptic neurons contributing to its activation. In this case the weights could still be learned in the same manner, but the postsynaptic neuron could only receive input from only one presynaptic neuron at a time. The weights in this case would be learned serially.

The weights can also be learned in parallel through an expansion of the above method. The postsynaptic neuron in this case is induced to fire at some time t_0 by way of an additional synapse with sufficient weight. The weights $w_{u_i,v}$ on the synapse from neuron u_i to neuron v are updated by the following rule

$$\Delta w_{u_i,v} = \eta(t_0 - t_{u_i})$$

where t_{u_i} is the firing time of neuron u_i . Further, the weight vector must be normalized after each application of the learning rule.

5 IMPLEMENTATION OF LEARNING

Our implementation of supervised learning with a spiking neuron, like that presented by Ruf and Schmitt, is based on Hebbian learning. That is, synaptic weights are increased when both the presynaptic neuron is active at the time that the postsynaptic neuron produces a spike. The weights in this case are initialized randomly in the range from 0 to 1. The rule for the change of weight $w_{u_i,v}$ from neuron u_i to neuron v whenever neuron v produces a spike is

$$\Delta w_{u_i,v} = \eta \varepsilon_u(t - t_u) \chi$$

where η is the learning rate, $\varepsilon_u(t - t_u)$ is the response of neuron u at the time, t , that neuron v fires, and χ is an indicator of whether the spiking of neuron v is desired at this time. If the spiking is not desired, χ is -1 , otherwise it is 1. In other words, the weight of the synapse from u to v will change by an amount proportional to the contribution of neuron u at the time of firing and will be positive if the spiking is desired and negative if not desired. If a weight

is increased above 1 or decreased below 0 it is set to 1 or 0 respectively. Over time, this rule leads to neuron v firing at desired times and remaining inactive at other times.

Because training in this way only occurs when the postsynaptic neuron fires it is crucial that the neuron always has a chance to fire, even when the initial weights for its synapses are very low. There are two ways to ensure that a neuron always has at least some chance of firing. The first is to have a dynamic threshold. When the neuron fires the threshold is increased by some value. Over periods of inactivity the threshold is decreased. So, if a neuron does not fire for an extended period of time the threshold will lower enough such that it begins firing and hence learning. The second method is to have a base firing rate that the neuron maintains even when it receives no activation. In this case also the neuron has a chance to learn periodically even though it may not be receiving enough activation due to low weights. This base firing rate exists is known to exist in biological neurons. In this case we chose to implement the first method, a dynamic threshold.

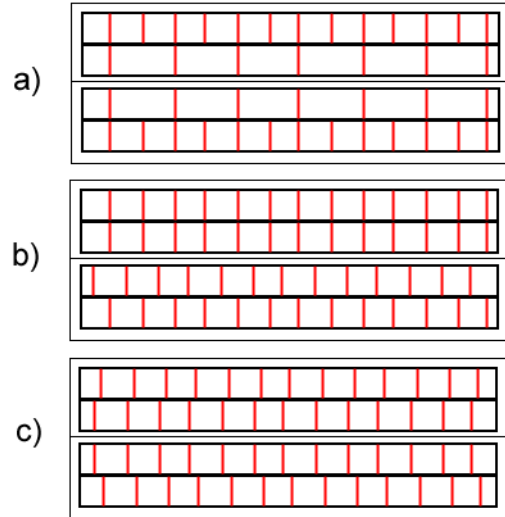


Figure 9: Three types of rhythms used in training. a) The top shows a high frequency spike train on the first input and a low frequency spike train on the second input. Below this is the opposite. Correspondingly, the first output neuron should spike for the top set and the second output neuron should spike for the bottom set. b) A training set where the inputs are either synchronized or not. c) A training set where either the first input regularly occurs shortly before or shortly after the second.

A set of simple test data was constructed to evaluate the learning rule above. Earlier generations of neurons, based on rate coding, often use simple functions of two inputs to test neural models and learning, such as the logical AND,

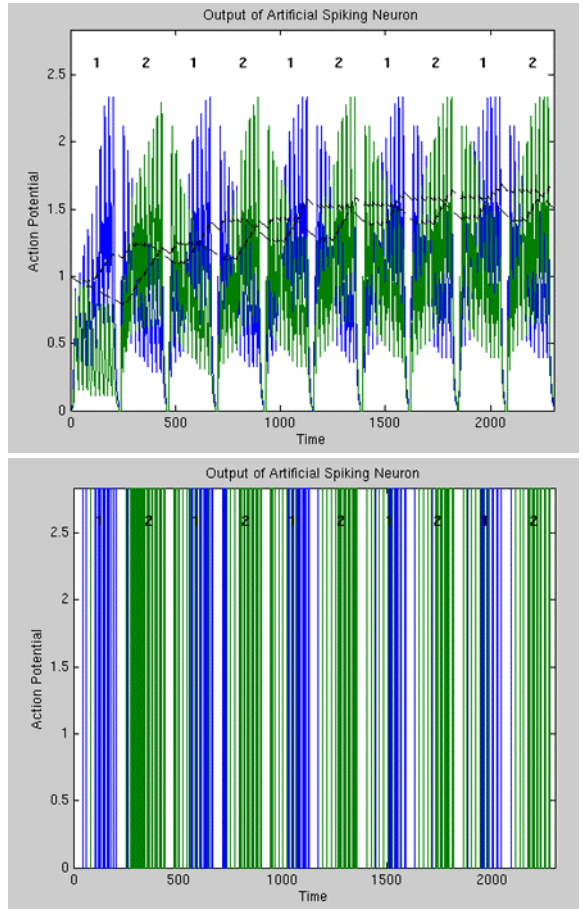


Figure 10: Top: Action potential of two output neurons while training on frequency rhythms. A “1” at the top of the graph signifies that a high frequency input is being given on input one and that output one (blue) should be more active. A “2” signifies that the opposite pattern is being presented and that the second output (green) should be more active. The dashed lines represent the dynamic thresholds of the output neurons. Bottom: The corresponding spikes of output one and two. A vertical line is one spike, blue for 1 and green for 2.

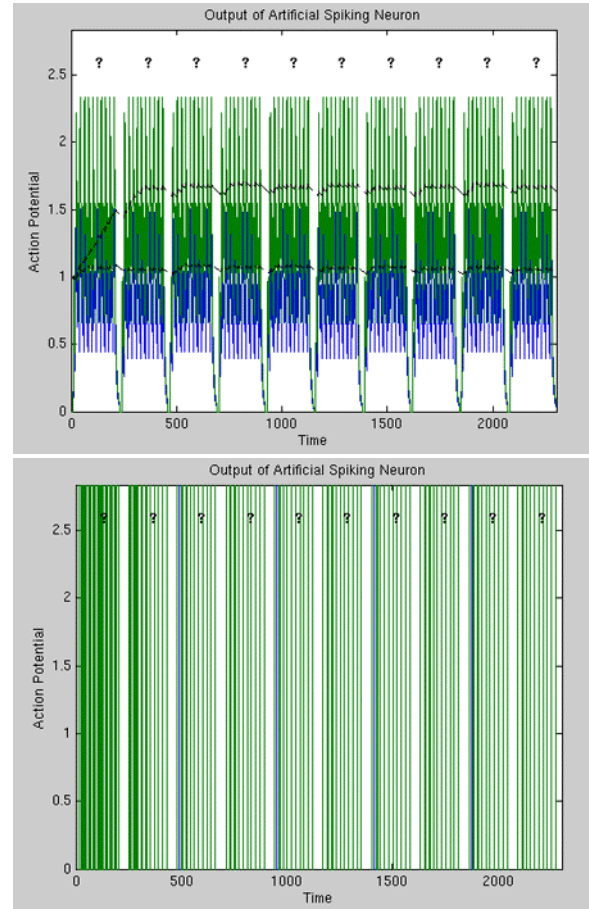


Figure 11: Top: Action potential of two output neurons while testing after training. Bottom: Spikes generated by output neurons. The question marks signify that the network does not know which output it is expected to provide.

OR and XOR functions. We sought to create equally simple functions based on temporal coding.

5.1 Rhythms

We developed three “rhythms” to test the presented supervised learning algorithm for spiking neurons. The networks in this case are composed of two input neurons, which receive the rhythmic sequences of firing, and two output neurons, that will fire to signal which rhythm is being presented. The three rhythms are based on the relative frequency of spikes, the synchrony of spikes, and the relative timing of spikes. These rhythms are presented graphically in figure 9.

5.2 Results

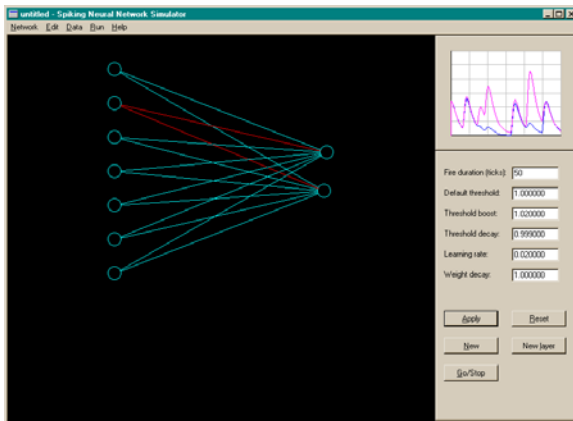
The graphs in figure 10 show the results of training a network of spiking neurons with two inputs and two outputs using the frequency based rhythms as training vectors. In the case of a high frequency spike train on the first input and a low frequency spike train on the second input the first output neuron should be active, that is, have a higher action potential and spike more often. In the case of the reverse the second neuron should be the more active.

You can see that after one a few presentations of data the appropriate neurons begin to fire when presented with the frequency patterns. While encouraging at first this result is actually misleading, as is shown by figure 11. These graphs show the response of the network after it has been trained and the weights are stationary. The network was presented with the same frequency rhythm data that it was trained on. You can see that one neuron is continually

active while the other hardly responds at all. The promising results during training were actually created by the changing of the weights and not the weights themselves. Each time a neuron is supposed to respond its weights are increased, hence making it respond more. Correspondingly, when it is desired that a neuron not fire its weights are decreased. This occurs each time that a pattern is presented. The network is small enough such that it can respond to these changes rapidly enough that it produces the correct responses for most of the time. Because the second neuron (graphed as green) was the last one to be trained positively during the learning sessions, it is the one that responds to the inputs in the testing set.

6 SPIKING NETWORK SIMULATOR

In order to experiment with and observe the dynamics of spiking nets, we created a Windows application which facilitates the creation of a pulsed neural network and provides rudimentary input and output resources. A screen shot is given below.

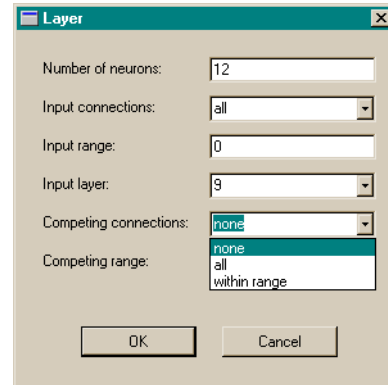


The simulation shown above was from a layer of seven input neurons to two outputs. The potentials of the two output neurons are recorded by the scrolling graph at the upper right.

Each input neuron may be fired by pressing an alphabetical key, where the letter “a” is mapped to the first neuron in the layer, “b” to the second, and so on. This was the chosen input method because (1) there needed to be a sensible way to interact with the input layer in real time, and (2) we were curious as to the capabilities of neural nets and especially spiking nets to recognize users by the rhythm of their typing.

Target outputs are chosen by pressing a numeric key “1” through the number of neurons in the output layer. This target was then trained while the simulator ran by our pseudo-Hebbian algorithm which will be described shortly.

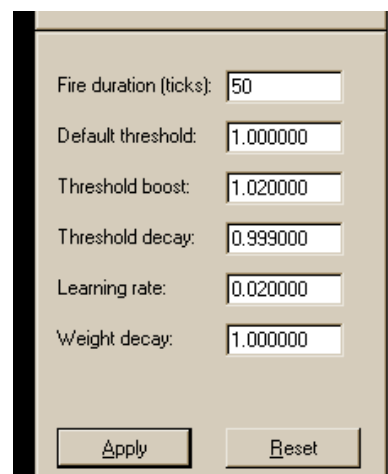
The networks that can be created on the simulator are highly configurable. Individual neurons may be added at any time and connected by dragging the mouse from the source neuron to the destination. New layers can be added en masse, with the options partially shown in this box:



Each layer can be connected to any of the previous layers where each neuron is connected to either all, none, or only those neurons which are in a range of the relative position of the destination neuron in its layer. Competing connections (connections within the same layer) can also be specified in the same manner. All competing connections are made inhibitory by default. This would allow us to experiment with Kohonen map-style self-organizing learning.

While training, neurons in the display window show their relative potentials as a shade of yellow and flash red when they fire. The coincidences of rhythm between neurons and the general activity level of each neuron is thus immediately visible.

Each of the dynamics of our learning model were configurable by the control panel on the right. The following shows our default experimental starting points:



The default threshold is given as a multiple of the maximum response function voltage. The threshold boost is a factor applied to neurons when they fire, partly to make up for their eventual loss of resistance through the periodic threshold decay, but partly to reduce their sensitivity when overstimulated. It is our observation that biological neurons consistently find a middle ground between not firing at all and firing at their maximum rate. Since neurons, like all cells, are essentially autonomous entities within the protection of the environment held constant by the body around them, it seems sensible that their gravitation toward the middle ground of firing (30-50Hz) is critical to their health. The learning rate is the η found in our learning equation,

$$w_{new} = w_{old} + \eta \epsilon_u(t - t_u) \chi$$

where t is the target value and ϵ represents the strength of the EPSP at the time of firing.

The simulator was used to verify the MATLAB results as per learning rhythms as well as to get immediate results with various experiments. Many configurations of network were tried and trained within a short period to gauge their abilities to learn. Though our nets did not realize a satisfying learning achievement, we ourselves learned to recognize many unpredicted dynamics of spiking nets through the observations of our data. The program is available at

<http://www.cs.rit.edu/~ccm2451/pub/snn.exe>

for anyone who is interested in using it.

7 REFERENCES

- 1 Gerstner, W. (1999) "Spiking Neurons". In W. Maass and C. M. Bishop (eds.), *Pulsed Neural Networks*. MIT Press, Cambridge, Mass., 1999.
- 2 Gerstner, W., Kempter, R., Leo van Hemmen, J., Wagner, H. (1999) "Hebbian Learning of Pulse Timing in the Barn Owl Auditory System". In W. Maass and C. M. Bishop (eds.), *Pulsed Neural Networks*. MIT Press, Cambridge, Mass., 1999.
- 3 Jahnke, A., Roth U., Schönauer, T. (1999) "Digital Simulation of Spiking Neural Networks". In W. Maass and C. M. Bishop (eds.), *Pulsed Neural Networks*. MIT Press, Cambridge, Mass., 1999.
- 4 Kohonen, T. (2001) *Self-Organizing Maps*. Springer Series in Information Sciences, Vol. 30, Springer, Berlin, Heidelberg, New York, 2001.
- 5 Krüger, J., Aiple, F. (1988) "Multielectrode investigation of monkey striate cortex: spike train correlations in the infragranular layers", *J. Neurophysiology*, vol. 60, pp 798--828, 1988.
- 6 Maass, W. (1997) "Networks of Spiking Neurons: The Third Generation of Neural Network Models" *Neural Networks*, 10(9):1659–1671.
- 7 Maass, W. (1999) "Computing with spiking neurons". In W. Maass and C. M. Bishop (eds.), *Pulsed Neural Networks*. MIT Press, Cambridge, Mass., 1999.
- 8 Ruf, B., Schmitt, M. (1997) "Learning temporally encoded patterns in networks of spiking neurons," *Neural Processing Letters*, vol. 5, no. 1, pp. 9--18, 1997.
- 9 Ruf, B., Schmitt, M. (1998) "Self-Organization of Spiking Neurons Using Action Potential Timing", in *IEEE Trans on Neural Networks*, Vol. 9, No. 3, May 1998, pp. 575-578.
- 10 Thorpe, S., Delorme, A., VanRullen, R. (2001) "Spike based strategies for rapid processing". *Neural Networks*, 14(6-7), 715-726.