Spiking Neural Networks and STDP-Based Remote Suprevision

Candidate number

1003559

A mini-project submitted for the degree of

Master of Mathematics in Mathematics and Statistics

Trinity 2019

Contents

1 Introduction

Recently, artificial neural networks (ANNs) have been remarkably successful in many fields including image classification [19, 14], object detection [24], natural language processing [16, 8] and many other applications in academia [31, 21]. Although the brain has historically been the inspiration for ANNs, they are still fundamentally different from the brain in terms of the elementary computing units and learning algorithms. Artificial neurons compute continuous values with a non-linear activation function and learn through backpropagation of gradients. However, neurons in the brain use discrete spikes (action potentials) to compute and transmit information [6]. Moreover, no evidence suggests that backpropagation is happening in our brain. Even the author of backpropagation, Geoffery Hinton, advocated to dispense this learning algorithm and start over [22].

To address the issue that artificial neurons are not entirely realistic, Maass [27] proposed spiking neural networks (SNNs) that closely resemble biological neural networks, where computation is performed on spikes. It has been shown that neurons in the SNNs have more expressive power than ordinary sigmoidal neurons (measured by VC dimensions) [27, 26]. Besides, spike time coding leads to a fast and extremely energy efficient neural computation, as in the brain [28]. Moreover, by understanding the 'language of neurons' (spike trains), it is possible to build brain-machine-interfaces that use SNNs to bridge the gap between neurons and computers [7, 36]. SNNs are also believed to be the stepping stone to understanding our brain and building an artificial general intelligence (AGI) [37].

On the other hand, the development of a biologically plausible supervised learning algorithm is challenging, as we do not know how exactly our brain learns with supervision. There have been a variety of learning algorithms that have been proposed in the last decades [3, 42, 38, 35]. We will focus on a biologically realistic remote supervised learning method proposed by Ponulak [38], which is based on an unsupervised learning mechanism that is observed in the brain.

In this report, we will discuss the basics of SNNs and the remote supervised method (ReSuMe). We will introduce some essential neuronal dynamics in Section 2 and an information encoding method in Section 3 as foundations to introduce SNNs in Section 4. Then we will derive the ReSuMe learning rule and show it can effectively train SNNs using results from the literature. In the end, we will discuss the constraints of SNNs and provide possible pathways for future research.

2 Neuronal dynamics

Neurons are the elementary processing units of our nervous system. A typical neuron has three functionally distinct parts: dendrite, soma, and axon. Intuitively, the dendrite acts as the 'input device' that collects signals from other neurons and transmits them to the soma. The soma plays the role of a 'central processing unit', where an important non-linear processing step is done: if the total input exceeds a threshold in a short interval of time, an output signal will be generated. The output signal is then delivered to other neurons via the 'output device'—the axon [10].

The junction between two neurons is called a synapse, where the axon (terminal) of the presynaptic neuron communicates with the dendrite (or soma) of the postsynaptic neuron. A single neuron in a vertebrate cortex often connects to more than 10^4 postsynaptic neurons [12].

2.1 Spiking neurons

Neuronal signals consist of short electrical pulses (also known as action potentials or spikes) that are generated and propagated by neurons due to the electrochemical properties of their cellular membranes [12, 10, 38]. The potential difference between the interior and exterior of a neuron cell is called the membrane potential. Without spike input, the neuron is at rest with a resting potential of about -65mV. After the arrival of a spike, the membrane potential changes. We denote the resting potential by $u_{\text{rest}} = -65\text{mV}$. Let $u_i(t)$ be the membrane potential of neuron i at time t. Suppose at $t = 0$, the presynaptic neuron j fires its spike and it is transmitted to the neuron i. For $t > 0$, the postsynaptic potential (PSP) is defined as

$$\epsilon_{ij}(t) = u_i(t) - u_{\text{rest}}.$$

If the voltage difference is positive, we have an excitatory postsynaptic potential (EPSP); otherwise, we have an inhibitory postsynaptic potential (IPSP) [10].

EPSP reduces the negative polarisation of the membrane potential and is known as depolarisation. Sufficient EPSPs within a short time interval can raise the membrane potential above a threshold, which triggers a spike. Figure 1 illustrates the change of membrane potential of the postsynaptic neuron with excitatory and inhibitory inputs. A single EPSP has an amplitude of about 1mV, and the threshold for spike initiation is usually 20mV to 30mV above the resting potential. In contrast to the schematic figure, most neurons need around 20-50 EPSPs to trigger a spike [12].

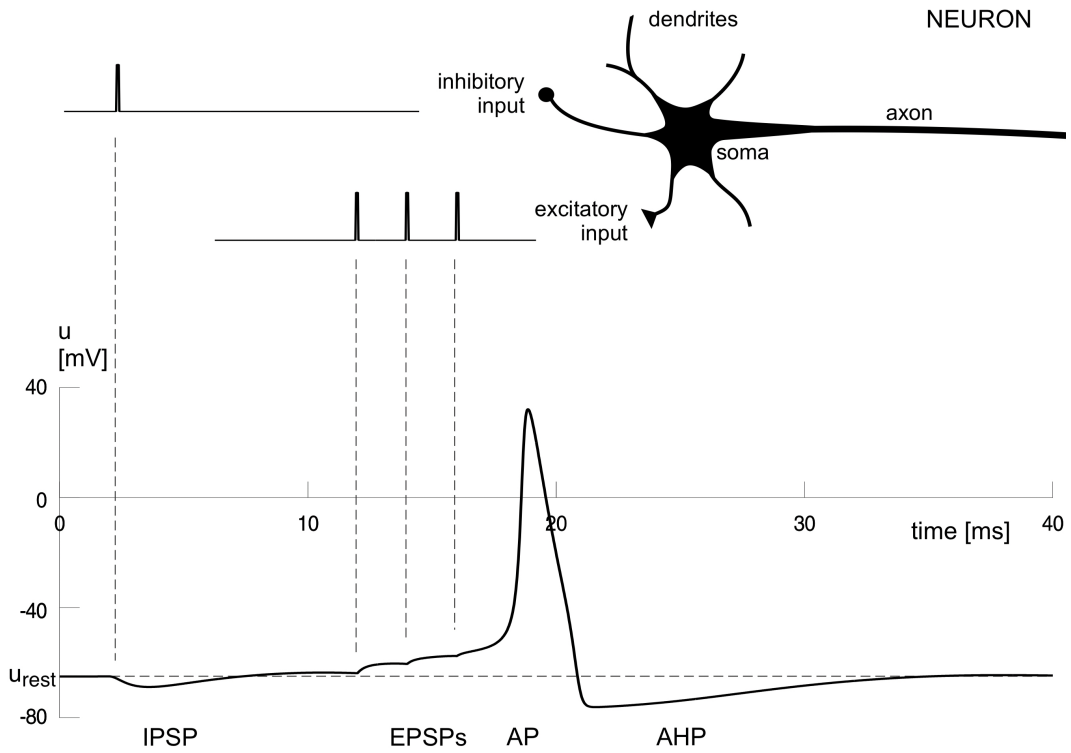


Figure 1: A schematic representation of a biological neuron (top) and the neuronal responses (IPSP and EPSP) to inputs from inhibitory and excitatory synapses (adapted from [38]). The summation of EPSPs triggers an action potential (AP) followed by an afterhyperpolarisation potential (AHP).

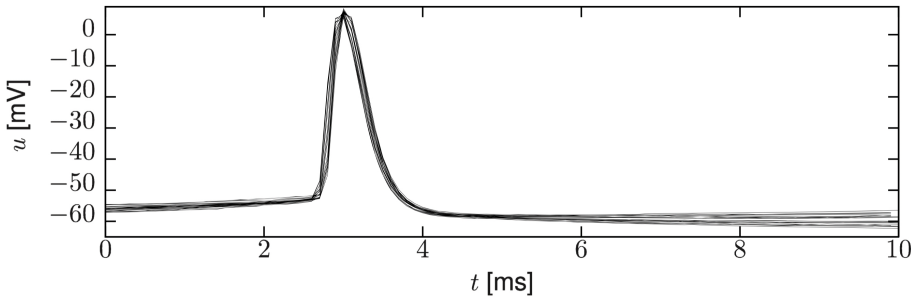


Figure 2: Membrane potential recordings u aligned by the time of maximum voltage show little variability of the shapes of action potentials (taken from [12]).

A chain of spikes emitted by a single neuron is called a spike train. The shape of isolated spikes from a single neuron are similar as shown in Figure 2. So the information is believed to be contained in the number and the timing of spikes rather than their shapes [12]. This motivates the mathematical definition of a spike train:

$$S_i(t) = \sum_f \delta\left(t - t_i^f\right) \tag{1}$$

where t_i^f for $f = 1, 2, \ldots$ denotes firing times of the neuron i, and $\delta(x)$ denotes the Dirac delta function with $\delta(x) = 0$ for $x \neq 0$ and $\int_{-\infty}^{\infty} \delta(x)\mathrm{d}x = 1$. Spikes in spike trains are usually well separated. After each spike, neurons have a refractory period when it is much more difficult to trigger new spikes. This is typically achieved by neurons repolarising to values below their resting potentials [33], as depicted in Figure 1.

2.2 The leaky integrate-and-fire model

In order to build a biologically realistic neural network, we need to describe the behaviour of neurons mathematically. One of the simplest and most popular models used to describe neuronal activities in SNNs is the leaky-integrate-and-fire (LIF) model. It begins with the idea that neurons can be simplified and viewed as RC circuits. The membrane acts as a capacitor to store charges coming into the cell, while some ion channels act as resistors slowly leak charges to the outside, as shown in Figure 3. We can, therefore, split the input current into two components, the current flowing through the resistor (I_R) and the current flowing through the capacitor (I_C)

$$\begin{aligned} I(t) &= I_R + I_C \\ &= \frac{u(t) - u_{\text{rest}}}{R} + C\frac{\mathrm{d}u}{\mathrm{d}t} \end{aligned}$$

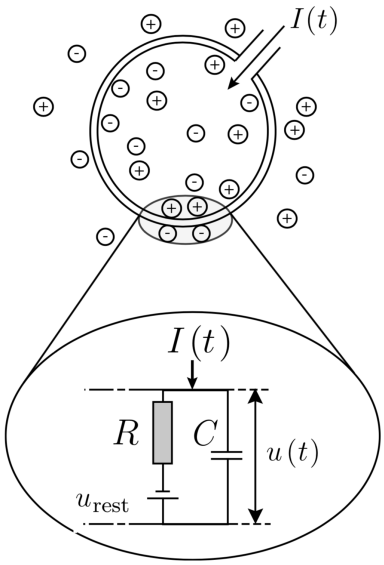


Figure 3: A schematic representation of a neuronal cell as an RC circuit (taken from [12]).

where we used the Ohm's law $I = U/R$ and the capacitive current $I_C = \mathrm{d}q/\mathrm{d}t = C\mathrm{d}u/\mathrm{d}t$. By rearranging and introducing a time constant $\tau = RC$ that controls the speed of which membrane potential falls back to its resting state, we have

$$\tau \frac{\mathrm{d}u}{\mathrm{d}t} = -[u(t) - u_{\mathrm{rest}}] + RI(t). \tag{2}$$

This linear ODE governs the dynamics of the membrane potential below the firing threshold.

We define the threshold condition for generating a spike in a neuron to be $u(t) = \vartheta$ (note this threshold might be different for different neurons), and the set of firing times are therefore defined as t^f such that $u(t^f) = \vartheta$. Moreover, as information is not contained in the shape of spikes, we do not care about the exact behaviour of action potentials. Instead, we treat spikes as instant events, which are triggered as soon as the membrane potential reaches the threshold, and reset the membrane potential to a voltage below the resting potential to mimic the afterhyperpolarisation potential. Formally, this is defined as

$$\text{if } u(t) = \vartheta, \quad \text{then} \lim_{\delta \to 0; \delta > 0} u(t + \delta) = u_r. \tag{3}$$

Equations (2) and (3) together define the LIF model. The reset process can also be formalised as removing a charge $Q = C(\vartheta - u_{\mathrm{r}})$ from the capacitor. Figure 4 demonstrates the behaviour of a spike and the reset process of the membrane potential described by the LIF model.

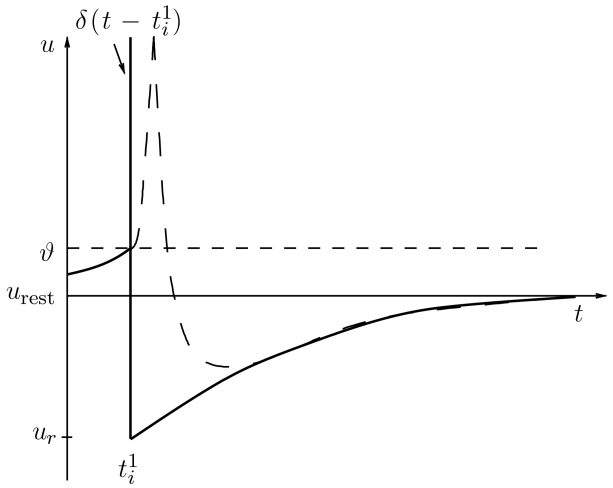


Figure 4: Illustration of a spike described by the LIF model (taken from [12]). The spike is replaced by a δ-pulse (the vertical line), and then the membrane potential is reset to u_r. The dashed line represents the shape of the actual action potential.

2.3 Synaptic current

Spikes generated by a presynaptic neuron are typically transformed into current pulses as the input to the postsynaptic neuron. The current flowing from neuron j to neuron i is defined as

$$I_{ij}(t) = g_{ij}(t)(E_{ij}^{\text{syn}} - u_i(t)) \tag{4}$$

where $g_{ij}(t)$ denotes the synaptic conductance and E_{ij}^{syn} denotes the reversal potential of the synapse. So $(E_{ij}^{\text{syn}} - u_i(t))$ is the potential difference between the synapse and the postsynaptic neuron across the cell membrane. For excitatory synapses, E^{syn} is approximately 0mV; whereas for inhibitory synapses, E^{syn} is around -75mV. The conductance $g_{ij}(t)$ is a function of time

$$g_{ij}(t) = \sum_f w_{ij} \exp\left(\frac{-t + t_j^f}{\tau_d}\right) H\left(t - t_j^f\right) \tag{5}$$

where τ_d is a time constant controlling the decay time, W_{ij} denotes the synaptic weight or efficacy, and $H(t - t_j^f) = 1$ for $t > t_j^f$ and 0 otherwise [18]. The synaptic efficacy depends on the activities of presynaptic and postsynaptic neurons, which is also known as synaptic plasticity and will be discussed in Section 5.

3 Neuronal coding

In order to process information in a biologically realistic way, we need to encode the data into the language of neurons—spike trains. Here we introduce population coding

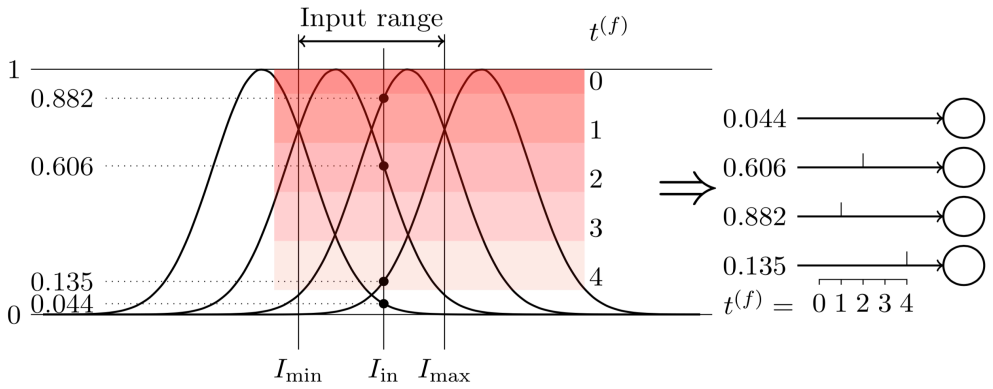


Figure 5: An array of Gaussian receptive fields for population coding (taken from [40]). Assign firing times $\{0, 1, 2, 3, 4\}$ to response values at $\{0.9, 0.7, 0.5, 0.3, 0.1\}$ respectively. The response values are rounded to the nearest integer firing times.

as used by Bohte et al. [3], which uses uniformly separated one-dimensional Gaussian receptive fields to cover the input data range. Each receptive field represents the sensitivity of a neuron to the input data, and each neuron is more sensitive to a specific range of data. Experiments, especially in the visual cortex [13], have shown that neurons generate spikes quicker for the range of data that they are more sensitive at. Suppose our numerical input values are in the range $[I_{min}, I_{max}]$. Bohte et al. [3] suggested arranging Gaussian receptive fields of m neurons centred at

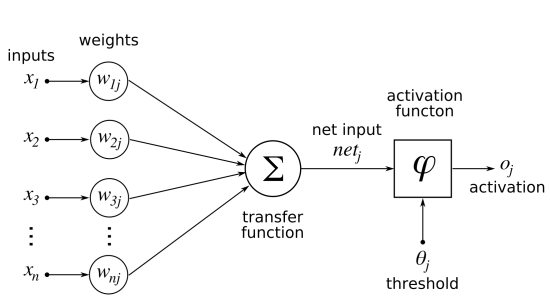
$$\mu_i = I_{\min} + \frac{2i-3}{2}\frac{I_{\max} - I_{\min}}{m-2}$$

where $i = 1, \ldots, m$. They share a common standard deviation defined as

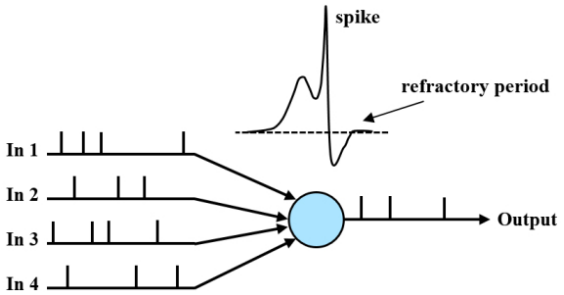
$$\sigma = \frac{1}{\beta}\frac{I_{\max} - I_{\min}}{m-2}.$$

Figure 5 shows the array of Gaussian receptive fields. For each input vector $I_{in} \in \mathbb{R}^n$, the response values of neurons encoding the respective variables were calculated, yielding $n \times m$ values in $[0, 1]$. To encode values into a temporal pattern of spike trains, we associate highly stimulated neurons with early firing times and less stimulated neurons with later or non-firing times. As experiments have shown spike times of a single neuron are noisy and exhibit 'jitters' of about 1-2ms [29], Bohte et al. [3] suggested assigning discrete integer values for spike times and round the responses to their nearest spike times as shown in Figure 5.

Following this method, we can encode any numerical input and output vectors into spike trains of a set of neurons, which accounted for regression tasks. On the other hand, for classification tasks, different neurons can be assigned with different classes and we can put each class at the centre of its corresponding neuron's receptive field to generate spike trains [3, 42].



(a) First and second generation artificial neuron (taken from [5]).



(b) The third generation artificial spiking neuron (taken from [23]).

Figure 6: Schematic diagrams of a typical neuron in the first and second generations of neural networks (left), and a neuron in the third generation of spiking neural networks (right).

4 Spiking neural networks

Maass [27] classified neural networks into three different generations according to their computational units. The first generation is based on the McCulloch-Pitts neuron, which is known as the first artificial neuron proposed in 1943 [34]. It calculates a weighted sum of inputs and passes them through a step function that outputs 0 or 1 based on a threshold θ. The second generation of neural networks is widely used today. Neurons aggregate inputs and pass them through a non-linear activation function (typically ReLU, sigmoid or tanh). Neurons in the first two generations take continuous numerical inputs and output either 0/1 or continuous values. Figure 6a gives a schematic illustration of neurons in the first two generations.

Spiking neural networks, on the other hand, is said to be the third generation of neural networks [27]. The main difference to previous generations is that neurons in SNNs compute on spikes, with input and output both being spike trains, which are binary events on a continuous temporal domain, as shown in Figure 6b. There are various models we can use to model the behaviours of spiking neurons. The most popular choice is the LIF model we introduced in Section 2, for its ease of computation. The input spike trains are aggregated and produce a synaptic current as defined in Equation (4), and the LIF model then simulates the output spike train. Note that the synaptic weights defined in Equation (5) is time-dependent, and may change over time by synaptic plasticity.

These artificial spiking neurons are used as building blocks for different SNN architectures. We will focus on feedforward SNNs in this report, where spike trains are only transmitted in one direction from input to output.

5 Learning

Recall the synaptic efficacy defined in Equation(5). If w_{ij} is constant, the response of the postsynaptic neuron i to a spike from the presynaptic neuron j will always be the same. However, electrophysiological experiments show that the change of postsynaptic potentials is not fixed but changes over time. This phenomenon is known as synaptic plasticity and is believed to be the mechanism of learning [18].

In 1949, D.O. Hebb proposed the first learning rule for neurons [15]. Hebb's rule stated: when a neuron j is near enough to excite neuron j and persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that j's efficiency in firing i is increased. This rule is often summarised as 'neurons that fire together wire together' [25]. Mathematically, Hebbian learning is expressed as

$$\Delta w_{ij} \propto \nu_i \nu_j \tag{6}$$

where ν_i and ν_j denote the firing rate of presynaptic neuron i and postsynaptic neuron j, and Δw_{ij} denote the change of the synaptic efficacy.

However, the original Hebb's rule did not consider synaptic depression (decreasing weights). If the synapses can only be strengthened, all weights will eventually saturate at maximum values. Followed from our definition for the synaptic current, we consider inputs with positive synaptic efficacies as excitatory, and inputs with negative synaptic efficacies as inhibitory. Moreover, the mathematical formulations of Hebbian learning depend on firing rates of neurons instead of precise firing times (spike trains), which contain more information.

5.1 Spike-timing-dependent-plasticity

Neurophysiological experiments suggested that the precise timing of spikes might influence the synaptic efficacy. The change in synaptic efficacies turns out to be a function of the difference in firing times of presynaptic and postsynaptic neurons [2, 32]. Presynaptic spike preceding the postsynaptic spike was observed to cause synaptic potentiation (strengthen), while the reverse order induced synaptic depression. This phenomenon is known as spike-time-dependent-plasticity (STDP). However, in some synapses, the complementary process was found. Synapses were depressed when the presynaptic spike was preceding the postsynaptic spike and strengthened otherwise. This is called anti-STDP. The STDP and anti-STDP processes are also known as Hebbian and anti-Hebbian processes [38].

Gerstner and Kistler proposed a general formulation describing synaptic plasticity [11]. The update rule for the STDP and anti-STDP processes between a presynaptic neuron i and postsynaptic neuron j is defined as

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}w_{ij}(t) =& S_j(t)\left[a_1^{\mathrm{pre}} + \int_0^\infty W_-(s)S_i(t-s)\mathrm{d}s\right] \\ &+ S_i(t)\left[a_1^{\mathrm{post}} + \int_0^\infty W_+(s)S_j(t-s)\mathrm{d}s\right] \end{aligned} \tag{7}$$

where $S_i(t)$ and $S_j(t)$ denote the spike trains of the presynaptic and postsynaptic neurons respectively; a_1^{pre} and a_1^{post} are non-Hebbian terms used for adjusting the average strength of the inputs to impose the desired level of activity; $W_-(s)$ and $W_+(s)$ denote the kernels which define the shapes of learning windows, with $s = (t_j^f - t_i^f)$ or $(t_i^f - t_j^f)$ being the difference between firing times of the presynaptic and postsynaptic neurons [10]. As there is no weight decay term in this formulation, the change in weights is considered to be long term—long term potentiation (LPD) or long term depression (LPD).

The learning window of the STDP process is often described as

$$W(s)^{\mathrm{STDP}} = \begin{cases} W_-(s) &= +A_- \cdot \exp\left(-s/\tau_-\right), \quad \text{if} \quad s \geq 0, \\ W_+(-s) &= -A_+ \cdot \exp\left(s/\tau_+\right), \quad \text{if} \quad s < 0, \end{cases} \tag{8}$$

where A_-, A_+ denote the amplitudes (positive for excitatory input, negative for inhibitory input); $\tau_-, \tau_+ > 0$ are the time constants determining the width of the learning window. On the other hand, the learning window of the anti-STDP process is defined as the opposite

$$W(s)^{\mathrm{aSTDP}} = \begin{cases} W_-(s) &= -A_- \cdot \exp\left(-s/\tau_-\right), \quad \text{if} \quad s \geq 0, \\ W_+(-s) &= +A_+ \cdot \exp\left(s/\tau_+\right), \quad \text{if} \quad s < 0, \end{cases} \tag{9}$$

with parameters defined similarly as above. Note that $W(s)^{\mathrm{aSTDP}} = -W(s)^{\mathrm{STDP}}$. The $(-/+)$ subscript indicate the spike from the presynaptic neuron i arrived earlier/later than that from the postsynaptic neuron j.

The learning window $W(s)$ for STDP is illustrated in Figure 7. Consider the first term in Equation (7): if we assume the postsynaptic neuron j fires at time t (otherwise this term has no contribution at time t), the integral calculates the convolution between the learning window $W_-(s)$ and the presynaptic spike train that arrived earlier than t. The weight change is positive and indicated by the right half in Figure 7. Now consider the second term in Equation (7): if we assume the presynaptic neuron i fires at time t, the integral calculates the convolution between the learning window $W_+(s)$ and the postsynaptic spike train that arrived earlier than t. The weight change, in this case, is negative and indicated by the left half in Figure 7.

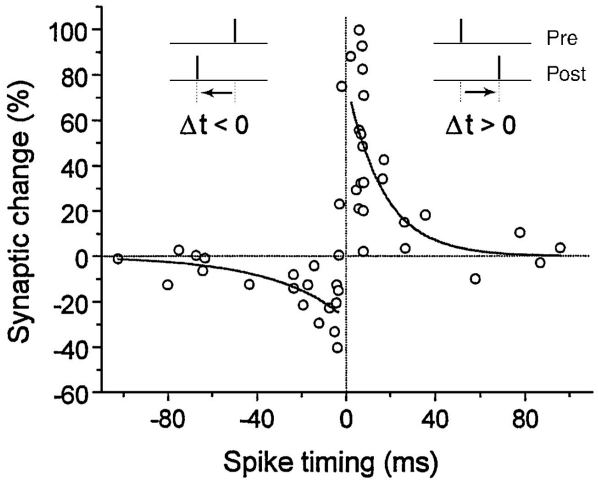


Figure 7: Illustration of the learning window $W(s)^{\text{STDP}}$ of the STDP process defined in Equation (8) (taken from [30]). Data points indicate the change of synaptic efficacy according to firing times of presynaptic and postsynaptic neurons, and are taken from experiments performed by Bi and Poo [2].

5.2 STDP-based remote supervision

Supervised learning in SNNs has been a challenge for a long time due to the binary-like signals and temporal information in spike trains. Bohte et al. [3] proposed the first supervised learning algorithm SpikeProp. However, the algorithm only works when information is encoded in the time of the first spike of each neuron instead of a spike train, and the backpropagation of the time derivatives is not biologically plausible. Ponulak [38] introduced another supervised learning method the remote supervised method (ReSuMe) that not only allows learning of spike trains but also based on STDP, which makes it more biologically plausible.

In this section, we consider a single layer SNN with an input layer and an output layer. The information is encoded in spike trains with input $S^{in}(t)$ and desired output $S^d(t)$. We have a set of input neurons $N^{in} = (n_1^{in}, n_2^{in}, ...)$ generating the input signals and a set of output or learning neurons $N^o = (n_1^o, n_2^o, ...)$ receiving signals from the learning neurons and are expected to generate the target spike trains. Moreover, Ponulak [38] introduced a set of teacher neurons $N^d = (n_1^d, n_2^{\text{d}}, ...)$ that are not part of the network, but remotely deliver the desired output S^d remotely into the network to supervise the learning process. Figure 8 shows the relationship between the three types of neurons. The remote supervision concept can be biologically justified by heterosynaptic plasticity, where the activity of a pair of pre- and postsynaptic neurons can affect other neurons that are inactive [9, 17].

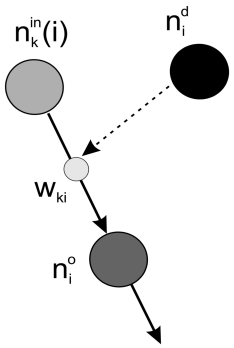


Figure 8: A schematic diagram of the input neuron $n_k^{in}(i)$, learning neuron n_i^o and the remote teaching neuron $n_i^d(i)$ (taken from [38]).

The basic supervised learning principle in artificial neural networks is to express the modifications to the weights by a function of error $e = y^d - y$ between the desired output y^d and the actual output y

$$\Delta w = f(e) = f\left(y^d - y\right).$$

The Widrow-Hoff learning rule [20] built upon this general formalisation and stated

$$\begin{aligned} \Delta w &= \left(y^d - y\right) x \\ &= y^d x - yx \end{aligned} \tag{10}$$

where x denotes the input.

Ponulak [38] interpreted the W-H learning rule using Hebbian process and formulated a supervised learning rule for SNNs. Recall the mathematical definition of the Hebbian process in Equation (6), and consider x, y and y^{d} as firing rates of the input, learning and teaching neurons. Then the first term in Equation (10) can be viewed as a STDP (Hebbian) process between the input neuron and the teaching neuron, and the second term with a minus sign can be viewed as an anti-STDP (anti-Hebbian) process between the input neuron and the learning neuron. This interpretation leads to the formulation of the ReSuMe learning rule

$$\frac{\mathrm{d}}{\mathrm{d}t} w(t) = S^d(t) \left[a^d + \int_0^\infty W^d\left(s^d\right) S^{in}\left(t - s^d\right) \mathrm{d}s^d \right] \tag{11}$$

$$+ S^o(t) \left[a^o + \int_0^\infty W^o\left(s^o\right) S^{in}\left(t - s^o\right) \mathrm{d}s^o \right] \tag{12}$$

where a^d and a^o denote the non-Hebbian terms, for excitatory synapses $a^d > 0$ and $a^o < 0$, whereas $a^d < 0$ and $a^o > 0$ for inhibitory synapses; $S^{in}(t), S^o(t)$ and $S^d(t)$ denote the input, output and teaching spike trains; W^d and W^o denote the learning

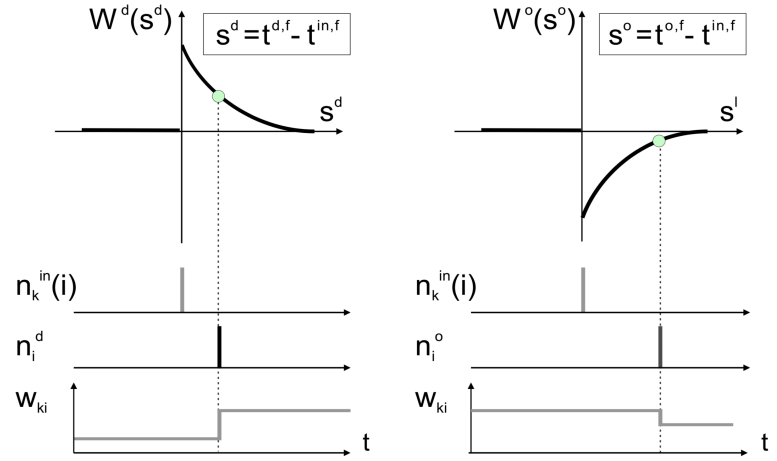


Figure 9: Illustration of the learning windows $W^d(s^d)$ (left) and $W^o(s^o)$ (right) (taken from [38]).

windows of the STDP and anti-STDP processes; $s^d = (t^{d,f} - t^{in,f})$ and $s^o = (t^{o,f} - t^{in,f})$ denote the delays between spikes from different pairs of neurons.

Simplified versions of learning windows are used to provide better results and faster convergence [38]. The learning window for the STDP process between the input neuron and the teaching neuron is defined as

$$W^d\left(s^d\right) = \begin{cases} +A_-^d \cdot \exp\left(\frac{-s^d}{\tau_-^d}\right), & \text{if} \quad s^d > 0, \\ 0, & \text{if} \quad s^d \leq 0. \end{cases}$$

The learning window for the anti-STDP process between the input neuron and the output neurons is defined as

$$W^o\left(s^o\right) = \begin{cases} -A_-^o \cdot \exp\left(\frac{-s^o}{\tau_-^o}\right), & \text{if} \quad s^o > 0, \\ 0, & \text{if} \quad s^o \leq 0, \end{cases}$$

where A_-^d and A_-^o determine the amplitude of response (positive for excitatory input, negative for inhibitory input); τ_-^d and τ_-^o denote the time constants controlling the width of learning windows. Figure 9 illustrates the learning windows defined above.

ReSuMe can be simplified by letting $a^d = -a^o = a$ and $A_+^d = A_+^o = A_+$. Then Equation (12) becomes

$$\frac{\mathrm{d}}{\mathrm{d}t} w(t) = \left[S^d(t) - S^o(t)\right] \left[a + \int_0^\infty W(s) S^{in}(t-s) \mathrm{d}s\right] \tag{13}$$

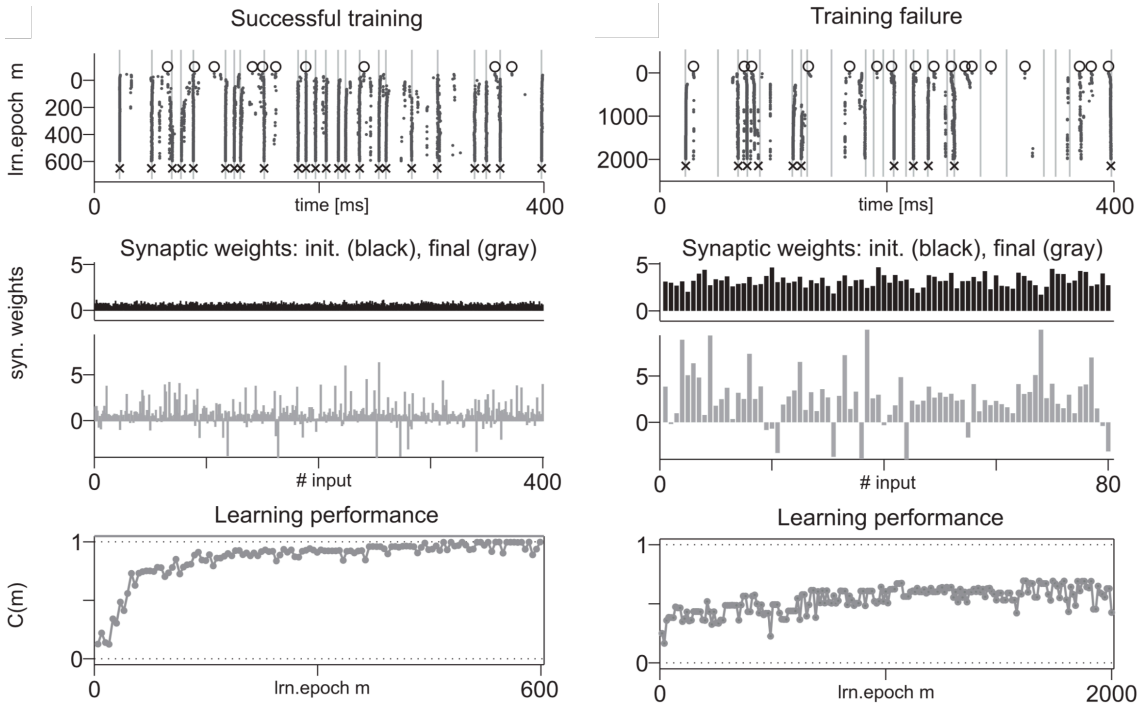
with learning window $W(s) = +A_- \exp\left(-s/\tau_-\right)$ for $s > 0$ and $W(s) = 0$ otherwise [38]. Besides biologically realistic, ReSuMe learn with spike trains (some learning algorithms can only learn with single spikes), and allows the use of any mathematical models describing the spiking neuron (here we use the LIF model) [17].

6 Results

ReSuMe is a temporally local algorithm—the weight update only depends on local learning windows. The global firing pattern is only to a limited extent relevant to update weights at any time. Despite this fact, results from the literature have proved that the algorithm is capable of learning arbitrary sequences of spikes in SNNs. Ponulak and Kasinski [39] performed several experiments on SNNs with a single output neuron to test the ability of ReSuMe. All experiments are conducted on the temporal domain 0-100ms.

For the first experiment, Ponulak and Kasinski generated a single target spike train S^d for the output neuron from a Poisson process with rate $r_o = 100$Hz, and input spike trains from Poisson processes with rate $r_i = 5$Hz for 400 input neurons. Synaptic weights are assumed to be positive initially and are initialised by a Gaussian distribution. Figure 10a shows the experimental results. The upper plot shows that the output spike train was quite different from the target spike train at the beginning of the training process. As the training started, many extra spikes appeared and almost covered all target spike times. This is because of a rise in synaptic weights from the small initialisation values. During training, the extra spikes gradually disappeared and the output spike train converged to the desired spike train. The middle plot shows the change in synaptic weights. We can see that some synaptic weights became negative after the training process, indicating inhibitory connections are needed in some synapses to suppress neurons from firing at undesired times. The bottom plot demonstrated a quantitative measure of the performance—$C(m) \in [0, 1]$—defined in [39] that measures the correlation between the output and target spike trains as a function of m epochs. The learning process is not particularly smooth, which is likely due to the locality of the learning algorithm.

In the second experiment, the number of input neurons was significantly reduced from 400 to 80 to test the ability of ReSuMe with limited input spikes further. To generate sufficient spikes at the beginning, synaptic weights are initialised to values higher than before. Figure 10b shows the results. In this case, the neuron failed to learn the exact target spike train, with the correlation index oscillating around 0.5 and failed to converge even for 2000 epochs. This oscillation is again due to the locality of the algorithm. The expressive power of the network is clearly not enough to generate the exact spiking pattern with such limited inputs. However, there are still about half of the target spikes that are learned by the network as shown in the



(a) 400 input neurons, synaptic weights are initialised according to a Gaussian distribution $\mathcal{N}(1 \times 10^{-10}, 5 \times 10^{-20})$.

(b) 80 input neurons, synaptic weights are initialised according to a Gaussian distribution $\mathcal{N}(3.5 \times 10^{-10}, 25 \times 10^{-20})$.

Figure 10: Experimental results of supervised learning with ReSuMe (taken from [39]). Each figure is divided into top, middle, and bottom. Top: firing times of the output neuron before training (circles), during training (dots) and after training (cross), with target firing times as grey lines. Middle: synaptic weights ($\times 10^{-10}$) before and after training, where input neurons are sorted chronologically according to their first spike time. Bottom: performance during training.

top figure. This demonstrates the ability of ReSuMe to make efficient use of available inputs even if these are extremely limited [39].

7 Discussion

In this report, we reviewed spiking neural networks from its foundation in neuronal dynamics and spike time coding. Then we reviewed the remotely supervised learning algorithm (ReSuMe) and derived it from the W-H rule and STDP. Experimental results confirmed ReSuMe is capable of learning arbitrary spiking patterns with enough input neurons. Researchers have also extended the algorithm to multilayer SNNs [41].

However, spiking networks with ReSuMe-like learning scheme have not yet outperformed the second generation neural networks or traditional machine learning

methods in simple classification tasks on the UCI dataset [1]. In fact, it is hard to find any experimental results where SNNs significantly outperformed traditional neural network models. There are a number of factors influencing the performance of SNNs: information encoding methods, network architectures, spiking neuron models and learning algorithms. Although ReSuMe is mostly biologically realistic, our brain does not have supervision for the exact targeted spike train. I believe we learn in a rather weakly supervised manner, and I think exploring biologically plausible adaptations of weakly supervised learning algorithms [43] in SNNs is a possible area for future research.

Another constraint for SNNs is the hardware—SNNs are more computationally intensive than ANNs as we need to calculate differential equations for each neuron to describe its behaviour. So it is not possible to build and train a huge SNN architecture as for computer vision and natural language processing. Neuromorphic computing hardware that is not based on the traditional Von Neumann architecture are now in development aiming to solve this problem [4]. I hope by introducing this neuromorphic hardware, SNNs can beat ANNs in performance and help us to gain a better understanding of the brain.

References

[1] Sami Abdul-Wahid. Spike-based classification of UCI datasets with multi-layer resume-like tempotron, 2018. URL `https://digitalcommons.cwu.edu/etd/1008`. accessed 20 April 2019.

[2] Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 18(24):10464–10472, 1998.

[3] Sander Bohte, Joost N. Kok, and Johannes A. La Poutré. Spikeprop: backpropagation for networks of spiking neurons. volume 48, pages 419–424, 2000.

[4] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. Spiking neural networks hardware implementations and challenges: A survey. *J. Emerg. Technol. Comput. Syst.*, 15(2):22:1–22:35, 2019.

[5] Chrislb. Diagram of an artificial neuron, 2005. URL `https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_english.png`. accessed 29 April 2019.

[6] Peter Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 2005.

[7] Julie Dethier, Paul Nuyujukian, Chris Eliasmith, Terrence C. Stewart, Shauki A. Elasaad, Krishna V Shenoy, and Kwabena A Boahen. A brain-machine interface operating with a real-time spiking neural network control algorithm. In *Advances in Neural Information Processing Systems 24*, pages 2213–2221. Curran Associates, Inc., 2011.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[9] Danielle Gerhard. Neuroscience. 5th edition. *The Yale Journal of Biology and Medicine*, 86(1):113–114.

[10] Wulfram Gerstner and Werner Kistler. *Spiking Neuron Models: An Introduction*. Cambridge University Press, New York, NY, USA, 2002.

[11] Wulfram Gerstner and Werner M. Kistler. Mathematical formulations of hebbian learning. *Biological Cybernetics*, 87(5):404–415, 2002.

[12] Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014.

[13] Tim Gollisch and Markus Meister. Rapid neural coding in the retina with relative spike latencies. *Science*, 319(5866):1108–1111, 2008.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[15] D.O. Hebb. *The Organization of Behavior: A Neuropsychological Approach*. John Wiley & Sons, 1949.

[16] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[17] Andrzej. Kasinski and Filip. Ponulak. Comparison of supervised learning methods for spike time coding in spiking neural networks. 2006.

[18] Christof Koch. *Biophysics of Computation: Information Processing in Single Neurons (Computational Neuroscience Series)*. Oxford University Press, Inc., New York, NY, USA, 2004.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105. Curran Associates Inc., 2012.

[20] Ben Krose, B Krose, Patrick van der Smagt, and Patrick Smagt. An introduction to neural networks. *J Comput Sci*, 48, 1993.

[21] Byunghan Lee, Seonwoo Min, and Sungroh Yoon. Deep learning in bioinformatics. *Briefings in Bioinformatics*, 18(5):851–869, 2016.

[22] Steve Levine. Artificial intelligence pioneer says we need to start over, 2017. URL `https://www.axios.com/artificial-intelligence-pioneer-says-we-need-to-start-over-1513305524-f619efbd-9db0-4947-a9b2-7a4c310a28fe.html`. accessed 28 April 2019.

[23] Jiaxing Liu, Hong Huo, Weitai Hu, and Tao Fang. Brain-inspired hierarchical spiking neural network using unsupervised stdp rule for image classification. In *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, ICMLC 2018, pages 230–235. ACM, 2018.

[24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[25] S Lowel and W Singer. Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity. *Science*, 255(5041):209–212, 1992.

[26] Wolfgang Maass. On the computational complexity of networks of spiking neurons. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 183–190. MIT Press, 1995.

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

[28] Wolfgang Maass. Computing with spikes. *Special Issue on Foundations of Information Processing of TELEMATIK*, 8, 2002.

[29] Wolfgang Maass and Christopher Bishop. *Pulsed Neural Networks*. MIT Press, 1998.

[30] Mojtaba Madadi Asl, Alireza Valizadeh, and Peter Tass. *Propagation Delays Determine the Effects of Synaptic Plasticity on the Structure and Dynamics of Neuronal Networks*. PhD thesis, 2018.

[31] Polina Mamoshina, Armando Vieira, Evgeny Putin, and Alex Zhavoronkov. Applications of deep learning in biomedicine. *Molecular Pharmaceutics*, 13(5):1445–1454, 2016.

[32] Henry Markram, Joachim Lubke, Michael Frotscher, and Bert Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, 275:213–5, 1997.

[33] G.G. Matthews. *Neurobiology: Molecules, Cells and Systems*. Wiley, 2000.

[34] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[35] AMMAR MOHEMMED, STEFAN SCHLIEBS, SATOSHI MATSUDA, and NIKOLA KASABOV. Span: Spike pattern association neuron for learning spatio-temporal spike patterns. *International Journal of Neural Systems*, 22 (04):1250012, 2012.

[36] A. N. Niranjani and M. Sivachitra. Motor imagery signal classification using spiking neural network. In *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, pages 901–904, 2017.

[37] Sidney Pontes-Filho and Stefano Nichele. Towards a framework for the evolution of artificial general intelligence. *CoRR*, abs/1903.10410, 2019.

[38] Filip Ponulak. *Supervised learning in spiking neural networks with ReSuMe method*. PhD thesis, 2006.

[39] Filip Ponulak and Andrzej Kasiński. Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting. *Neural computation*, 22(2):467–510, 2010.

[40] Sumit Shrestha. *Supervised Learning in Multilayer Spiking Neural Network*. PhD thesis, 2017.

[41] I. Sporea and A. Gruning. Supervised learning in multilayer spiking neural networks. *Neural Computation*, 25(2):473–509, Feb 2013. ISSN 0899-7667. doi: 10.1162/NECO_a_00396.

[42] Qiang Yu, Huajin Tang, Kay Chen Tan, and Haizhou Li. Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns. *PLOS ONE*, 8:1–16, 2013.

[43] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2017.