

Oscar T. Skaug

# Spiking Neural Networks: A Survey

Bachelor's project in Computer engineering

Supervisor: Ole Christian Eidheim

June 2020

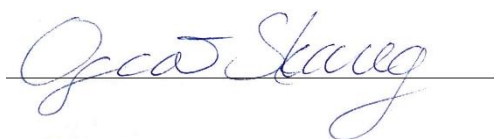


## Abstract

The purpose of the survey is to support teaching and research in the department of Computer Science at the university, through fundamental research in recent advances in the machine learning field. The survey assumes the reader have some basic knowledge in machine learning, however, no prior knowledge on the topic of spiking neural networks is assumed. The aim of the survey is to introduce the reader to artificial spiking neural networks by going through the following: what constitutes and distinguish an artificial spiking neural network from other artificial neural networks; how to interpret the output signals of spiking neural networks; How supervised and unsupervised learning can be achieved in spiking neural networks; Some benefits of spiking neural networks compared to other artificial neural networks, followed by challenges specifically associated with spiking neural networks. By doing so the survey attempts to answer the question of why spiking neural networks have not been widely implemented, despite being both more computationally powerful and biologically plausible than other artificial neural networks currently available.

Norwegian University of Science and Technology

Trondheim, June 2020

A handwritten signature in blue ink, reading "Oscar T. Skaug", written over a horizontal line.

Oscar T. Skaug

## The bachelor thesis

The title of the bachelor thesis is “Study of recent advances in artificial and biological neuron models” with the purpose of “supporting teaching and research in the department of Computer Science at the university through fundamental research in recent advances in the machine learning field”.

Due to the nature of the thesis, the student gained a lot of freedom in how he wanted to approach the task. The student had completed the subject TDAT3025 “Applied machine learning with project” the previous semester (autumn 2019). TDAT3025 require the student to complete a project, and the student wrote a review article on biological neuron models for the project. The student became interested in biological neuron models and mathematical models for neurons during the project and was therefore driven towards research on the topic of biologically plausible spiking neuron models. Spiking neural networks, which is a network of spiking neurons was a prime target for a research topic for the survey, as the “newest” addition of artificial neural networks in terms of implemented neuron models. Hence this survey came to, as a natural extension of the project the student conducted the previous semester. The student came in closer contact with the teacher of TDAT3025 during the project and started planning together with the teacher for a bachelor thesis undertaking, supervised by said teacher. Note, the project the previous semester only served as a motivation for studying artificial neural models and is not required or assumed to have been read for this thesis and is neither referenced as a source.

## Contents:

<b>1. Introduction</b>	1
1.1 Research Questions	2
1.2 Thesis Organization	3
<b>2. From Biological Neurons to Artificial Neural Networks</b>	5
2.1 The Biological Neuron	5
2.2 Traditional Neuron Models	6
2.2.1 First Generation of Artificial Neural Networks	8
2.2.2 Second Generation of Artificial Neural Networks	11
2.3 Third Generation of Artificial Neural Networks	12
2.3.1 Spiking Neuron Models	13
2.3.1.1 The Hodgkin-Huxley Model	14
2.3.1.2 The Integrate and Fire Model	15
2.3.1.3 The Izhikevich Model	17
2.3.2 Spiking Neural Networks: A Network of Spiking Neurons	18
<b>3. Neural Coding</b>	20
3.1 Rate Coding	20
3.2 Spike Coding	21
<b>4. Learning in Spiking Neural Networks</b>	24
4.1 Supervised Learning	24
4.2 Unsupervised Learning	26
4.3 DNN-to-SNN Conversion	28
4.4 Performance of Spiking Neural Networks	29
<b>5. Benefits of Spiking Neural Networks</b>	31
5.1 Brain Similarity	31
5.2 Computational Power	32
5.3 Energy Efficiency	34
5.3.1 Pseudo Simultaneous Input-Outputs	35
<b>6. Challenges of Spiking Neural Networks</b>	37
6.1 The State of Spiking Neural Networks	37
6.2 Comparability	38
6.3 Non-Optimal Hardware	38
<b>7. Conclusion</b>	39
7.1 Further Work	40
<b>8. References</b>	42

## Acknowledgements

I would like to extend my outmost gratitude to my supervisor Ole Christian Eidheim for giving constructive feedback as well as being supportive throughout my bachelor thesi

# 1. Introduction

Humans have throughout time taken inspiration from living organisms, including our own bodies, when inventing and creating new things. Inspiration include how humans in the early days mimicked how other animals hunted, to newer mechanical designs, such as bullet trains, taking inspiration from bird beaks to reduce air resistance, thereby reducing energy consumption. The field of artificial neural networks are no different, which can be viewed as an attempt to recreate parts of the biological neural network. After all, artificial neural networks were a direct result from researchers trying to mimic how the biological neural network is structured and function. Artificial neural networks implement artificial neurons and synapses to both learn and perform a variety of tasks. artificial neural networks see widespread implementation as of today, being used in tasks such as, classification, translation, recognition, filtering and more.

The field of artificial neural networks was in no small part pioneered by neurophysiologist Warren McCulloch and mathematician Walter Pitts, in their paper “A logical calculus of the ideas immanent in nervous activity” (1943) [64]. In relation to their paper, they created a simple neural network using electrical circuits to demonstrate how the neurons in the brain might operate on simple logical operators. There had already existed a community of scientists performing mathematical work on neural networks when McCulloch and Pitts published their paper. The new ideas brought forth by McCulloch and Pitts was how they used propositional logic and computation to explain neural events. The idea of McCulloch and Pitts, explaining neural activity through logic gates, became the foundation for boolean neuron models and a network of such neuron models – the first-generation artificial neural networks. It is therefore understandable, that McCulloch and Pitts’ paper are often cited as the starting point in artificial neural network research.

Now more than half a century has passed since the McCulloch and Pitts paper and the invention of, what would later be known as the “first-generation” artificial neural networks (1958) [43]. Since then, the field of artificial neural networks have seen both periods of rapid progress and periods of almost standstill. In recent times, the relatively newly success of deep neural networks has made the field of artificial neural networks receive a revived interest, both in the form of funding and amount of research [22]. The newest generation of artificial neural networks, the third generation, is known as spiking neural networks. Although spiking neural networks are known as the latest generation of artificial neural networks, they are still not widely used till date. The idea of spiking neuron models, the neuron model used in spiking neural networks, is not a new concept. The earliest spiking neuron model, the Hodgkin-Huxley model (1952), have been around for longer than the first-generation artificial neural networks. However, implementing spiking neurons in artificial neural networks are still quite a novel idea compared to other neuron model implementations. Spiking neural networks has experienced an increased interest as a research topic the last couple of years, as they offer an unprecedented amount of biological

plausibility, compared to earlier neuron models. There is no consensus, however, as to the future outlook for spiking neural networks. While some argue that commercialized implementations of spiking neural networks are right around the corner, others are sceptic about the implementation potential. The scepticism is often related to the lack of commercial implementations of spiking neural networks, even after several decades since they were first introduced.

This survey aims to introduce the reader to spiking neural networks. The first topic of the survey is the differences between spiking neural networks and traditional artificial neural networks. The rest of the survey will focus on certain aspects of spiking neural networks, including information processing, learning, and some of the benefits and challenges associated specifically with spiking neural networks. By doing so, the survey attempts to answer the research question introduced in chapter 1.1.

## 1.1. Research Questions

The topics included in this survey has been chosen to form a basis for answering the following two research questions:

Research question 1: Will spiking neural networks replace sigmoidal neural networks the same way sigmoidal neural networks replaced its predecessor?

The first-generation artificial neural network saw itself become essentially replaced by the second-generation artificial neural network, the sigmoidal neural network. It is natural to assume that a newer generation will replace its predecessor, which was the case when the sigmoidal neural network was introduced. However, spiking neural networks, the third-generation artificial neural network, have yet to replace the sigmoidal neural network, despite having been around for several decades already. Hence, the first research question raises the question, can spiking neural networks be expected to replace sigmoidal neural networks eventually, or will spiking neural networks never replace the sigmoidal neural networks.

Research question 2: If spiking neural networks are set to replace sigmoidal neural networks, what have held them back from doing it so far?

The second research question can be seen as an extension of the first research question. If spiking neural networks are set to replace sigmoidal neural network, why has it not occurred yet?



## 1.2. Thesis Organization

In order to answer the research questions given in chapter 1.1, it is necessary to explore several topics concerning spiking neural networks. To explore said topics, the chapters are organized as in the contents list. This sub-chapter aim to give a short summary of all the main chapters included in the survey:

### 1. Introduction

Chapter 1 serve as an introduction and aims to give the reader an overview of what the survey will contain. Further, the research questions, that guided the investigation of the research conducted for the survey, is presented.

### 2. From Biological Neurons to Artificial Neural Networks

Chapter 2 starts with introducing the biological neuron, as the basis from which artificial neurons are modelled after. The McCulloch and Pitts neuron model will be introduced, followed by the artificial neuron model used for both the first- and second-generation neural networks. The chapter will then introduce the reader to spiking neuron models, used by the third-generation artificial neural networks, spiking neural networks. From reading chapter 2, what constitutes the three generations of artificial neural networks, and what differentiates them from one another, should become clear.

### 3. Neural Coding

Chapter 3 concerns how to interpret the output signals from spiking neurons. The output of spiking neuron models is not a single value, but instead a time window with distinct spike-events. Spiking neuron models cannot make use of the same interpretation methods as traditional neuron models, as spiking neuron models include the temporal axis in the input and output. There exist several different methods of interpreting the output of artificial spiking neurons, which all stem from the field of neuroscience. Chapter 3 will introduce some of the most common interpretation methods implemented by spiking neural networks; rate-coding, time to first spike coding, rank order coding and latency coding.

### 4. Learning in Spiking Neural Networks

Chapter 4 is dedicated to methods of training spiking neural networks. Both supervised and unsupervised learning will be discussed. Further a way of training a spiking neural network, by converting an already trained traditional deep neural network to a spiking neural network is also included in the chapter. Lastly, the performance results of spiking neural networks will be discussed, by referring to accuracy results achieved on the benchmark dataset MNIST.

## 5. Benefits of Spiking Neural Networks

Chapter 5 mention some of the benefits of spiking neural networks compared to other artificial neural networks. The chapter will discuss the benefit of the gained biological plausibility of spiking neural networks, compared to other artificial neural networks. Further, the chapter will mention some of the investigation work conducted by Wolfgang Maass on the topic of the computational abilities of spiking neurons. The chapter will lastly mention how spiking neural networks can be more energy efficient, compared to other artificial neural networks. The energy efficiency gain of spiking neural network will be mentioned in relation to neuromorphic hardware.

## 6. Challenges of Spiking Neural Networks

Chapter 6 aim to summarise the challenges specifically associated with spiking neural networks. Several of the challenges of spiking neural networks will already have become apparent though the earlier chapters. Chapter 6 therefore, aim to focus on the challenges explored in less detail from other chapters. The challenges that will be discussed in more detail, is the titles of the sub-chapters of chapter 6. The unclear state of spiking neural network considers the fact that not all agree on the definition of spiking neural networks. The different definitions used can cause huge differences in how prominent figures in the field of AI view the potential of spiking neural networks. The next sub-chapter is dedicated to the benchmarks used for measuring the performance of artificial neural networks. Generally, the benchmarks used for measuring performance of artificial neural networks was not created with spiking neural networks in mind. Hence, the benchmarks often fail to showcase most of the benefits associated with spiking neural networks. Lastly, a short sub-chapter mention the lack of available neuromorphic hardware, which is needed for many of the benefits for spiking neural network implementation.

## 7. Conclusion

Chapter 7 try to make some conclusions about the future of spiking neural network by answering the research questions presented in chapter 1.1. Further, future potential areas of investigation not included in the survey, is mentioned. The future work can also serve as a starting point for further reading by the reader. The topics included as potential future work is: reinforcement learning in spiking neural networks, reservoir computing and the spike response model as a spiking neuron model.

## 2. From Biological Neurons to Artificial Neural Networks

To explain what constitutes a spiking neural network, it is necessary to mention the other types of artificial neural networks. Further the author believes it to be constructive to start with a short introduction of the biological neuron, as it serves as a reference point for artificial neuron modelling. This chapter attempts to give the reader the information necessary to distinguish between what is referred to as the 1<sup>st</sup>, 2<sup>nd</sup>, and 3<sup>rd</sup> generation artificial neural networks. By doing so, some of the strengths and weaknesses associated with each generation of artificial neural network might become apparent.

### 2.1. The Biological Neuron

Our understanding of the biological neuron has changed greatly over time. From believing that the biological neural network was one single continuous network, as proposed in the Reticular theory in the 19<sup>th</sup> century, to much newer research aimed at gaining a better understanding of each individual component of the neuron. Hence, our understanding of the neuron is constantly updated with the advent of new research results. New research might give insight in some of the current mysteries of the biological neural network, such as how they learn, allow for the formation of thoughts, and make decisions. New discoveries might have a crucial impact on how artificial neural networks will be created in the future. Nevertheless, a short summary of the current understanding of how biological neurons communicate and form networks is given, as to give a rough idea of the understanding from which the current artificial neurons are modelled after.

A real neuron generally consists of the soma (The cell body), a set of dendrites and one single axon. An action potential – a neural spike – is generated in the soma, serving as the signal transmitting information between neurons. This action potential, which is a short electrical pulse, travels down the axon of the neuron until it reaches what is called the synapse. The synapse connects one neuron to another by connecting the axon of the firing neuron, to a dendrite of the receiving neuron. The axon end of the synapse receives the action potential and converts it to a chemical signal, in the form of neurotransmitters, that travel across the synaptic cleft. The neurotransmitters then bind to receptors on the dendritic end of the synapse. These receptors influence the membrane potential of the receiving neuron by opening and closing ion channels, allowing different types of ions to flow in and out of the membrane. Different neurotransmitters can both decrease (inhibitory transmitter) and increase (excitatory transmitter) the membrane potential of the receiving neuron. The excited or inhibited membrane of the receiving neuron will have its membrane potential gradually go towards its resting potential. If the incoming excitatory signal makes the membrane potential reach a certain threshold however, a spike is formed by the neuron that received the excitatory signal, sharply decreasing the membrane potential in the

process. The newly created spike will then in turn be passed on to another neuron repeating the steps in the earlier explanation.

Several neurons can send an action potential to one receiving neuron, each influencing the membrane potential of the receiving neuron. Therefore, it is important to note that the membrane potential change with time, such that if a neuron fires a signal at  $t_{initial}$ , and another at  $t_{initial} + 0.1ms$ , they will both influence the receiving neuron. The signal from the two neurons will together decide if the receiving neuron will fire. However, the membrane potential generated from the first neuron would already have started to change. It is also important to note that the axon of a neuron can be connected to several other neurons through different axon terminals and can even be connected to dendrites of its own neuron (called an autapse).

The “passageways” for information, the links between neurons consisting of axon, synapse and dendrite may change over time, facilitating either a strengthening or weakening of the relationship between the firing of one neuron and the generation of a spike in the receiving neuron. The biological neural network is also able to both form new links between neurons that did not have a link already, as well as remove links between neurons that see sparse to no spike transmission. The changing in the strength of the link between neurons and the formation of new links is what is theorized to be the underlining principle behind learning in biological neural networks. However, what decides when and how much a link is strengthened or formed is still a debated topic. Most agree however, that whatever decides the change in the links between neurons, will likely vary depending on the area, as well as the current development of the brain in question.

As stated before, the explanation above is meant to be rough and simplified, thereby possibly omitting details that might impact neural computation. The explanation of the neuron drew the picture of axons, dendrites and synapses to be nothing more than “passive” passageways, with their only function being transmitting a spike from the soma of one neuron, to that of another. However, it is known that dendrites have a significantly more active role in the formation of spikes [17]. Nevertheless, the given explanation should suffice to serve as a basis of artificial neural models. The reason being that all neuron models implemented by artificial neural networks are significantly simplified, even more so than the given explanation of the biological neuron. The reason for the simplification in artificial neuron models, is to reduce the computational cost of simulating the neurons.

## 2.2. Traditional Neuron Models

In the field of machine Learning, the terms 1<sup>st</sup>, 2<sup>nd</sup>, and 3<sup>rd</sup> generation neural networks are often used. However, the definition of what constitutes the different generations of neural networks is not unanimously agreed upon. Where the line is drawn between the different generations is somewhat cloudy, especially so for the 1<sup>st</sup> and 2<sup>nd</sup> generation, as both use the same neuron model to form neural networks. This survey will use the definitions outlined by

Maass in “Networks of spiking neurons: The third generation of neural network models” [55], for the different generations of artificial neural networks.

As mentioned in the introduction, the neural network designed by McCulloch and Pitts was the predecessor for the first generation of artificial neural networks. The neural network of McCulloch and Pitts consisted of artificial binary neurons, henceforth referred to as “McCulloch and Pitts neurons”. The McCulloch and Pitts neuron would take the sum add the incoming signals together, before inserting the summed value into a threshold function. If the value reached a given threshold, the output of threshold function would be set to 1, representing the McCulloch and Pitts neuron in question being “active”, as in firing a spike. If the value of the sum given to the threshold function failed to reach the threshold, the function would return 0, representing the McCulloch and Pitts neuron staying “inactive”, meaning no spike was formed. The reasoning for interpreting the output of the McCulloch and Pitts neuron as a boolean value, 1 or 0, came from an observation of the biological neuron: at a given time, a biological neuron either fires a spike, or not. It is not possible for a biological neuron to fire say half a spike. Hence the only states a biological neuron could take at any given moment was that of “firing a spike” or “not firing a spike”. Therefore, as the state of a biological neuron only had two options regarding spike firing, it could be represented by a boolean value.

The McCulloch and Pitts neuron, receiving input from other neurons and sending out its own output  $y$ , is visualized in fig.1:

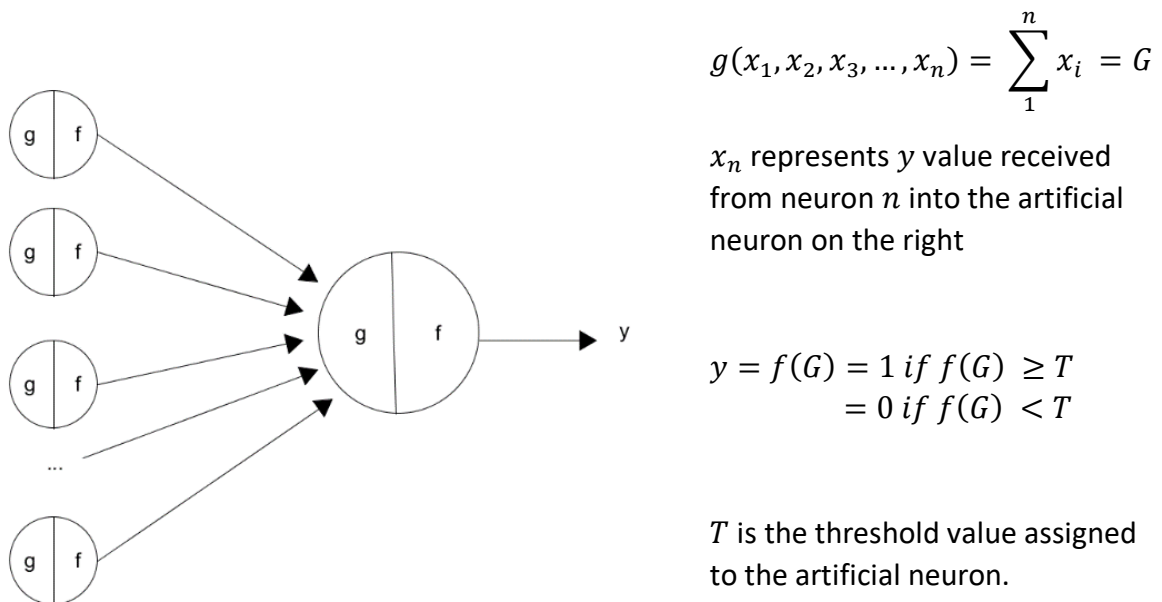


Fig.1: McCulloch and Pitts neuron model

The function  $f(G)$  in fig.1 served as the threshold function, mapping the sum of inputs to a boolean output  $y \in \{0,1\}$ .

The McCulloch and Pitts neuron were able to make a decision based on a set of binary inputs and could represent several Boolean functions such as AND, OR, NOR and NOT by assigning different values to  $T$  and changing the sum operation of the inputs. An example of an AND function with four inputs, requiring the inputs from  $x_1$  through  $x_3$ , with  $x_4$  as an inhibitory input, can be represented by the following threshold, and sum operation given in eq.1. The given neuron would only fire if the inputs from  $x_1$  through  $x_3$  was all 1 and  $x_4$  was 0:

$$g(x_1, x_2, x_3, x_4) = -x_4 + \sum_{i=1}^3 x_i = G, \quad T = 3 \quad (\text{eq. 1})$$

Although a network of McCulloch and Pitts neurons could be modelled as to perform several different boolean operations, it was unable to learn. The inability to learn came as a consequence of two flaws with the McCulloch and Pitts neuron model. The biggest factor for the inability to learn was due to McCulloch and Pitts neurons lacking a way of representing the strength of the links (henceforth referred to as the “weight”) between neurons. Since the model lacked a representation for weights, it was unable to assign different importance to given inputs. Further, as mentioned in chapter 2.1, it is the changing of weights that allow a biological neural network to learn. The second reason for the McCulloch and Pitts network being unable to learn was the fact that both the threshold value  $T$ , and sum operation  $g(x_1, x_2, x_3, \dots, x_n)$ , had to be hard coded, for all neurons in the network. Since the threshold value and sum operation had to be hardcoded, it became impossible for the network itself to change them given training data.

### 2.2.1. First Generation of Artificial Neural Networks

The first-generation artificial neural networks came with the invention of the perceptron neuron model, proposed by Frank Rosenblatt in his paper “The perceptron: a probabilistic model for information storage and organization in the brain” [43]. The perceptron model would later be refined by Minsky and Papert in 1969, resulting in the perceptron model as it is known today. The perceptron neuron model solved the biggest problem of the McCulloch and Pitts neuron, namely the lack of weights.

The first-generation artificial neural networks were defined by Maass[55] as; a network of perceptron neuron models implementing threshold functions. The output signal from perceptron neurons in first-generation artificial neural networks therefore continued to be a boolean output  $\{1, 0\}$ . However, the boolean output from the perceptron neuron would be multiplied with a weight  $w_i$ , before being given as the input to another perceptron neuron in the network. Therefore, the input values for the perceptron neurons could be any real value  $\mathbb{R}$ . The real valued inputs would still be added together, before being put through the

threshold function, the same way as in the McCulloch and Pitts neuron models. However, the way a perceptron neuron received its threshold value and used it, changed from the McCulloch and Pitts neuron. Instead of the threshold value being hard coded, the perceptron model received its threshold value as a separate “threshold input”. The threshold value  $T$  was represented by a weight  $w_0 = -T$ . For the perceptron to know the threshold, the weight  $w_0$  had to be multiplied with an input signal  $x_0$  resulting in the threshold input. The input signal  $x_0$  was set to always equal 1, so the perceptron would receive the threshold input every time it received inputs. The threshold input would behave the same as other inputs and be added together with the rest of the inputs resulting in a value  $G$ . However, the value of the threshold input was  $-T$ , not the threshold value itself. Therefore,  $G$  is effectively the sum of the inputs,  $x_1, x_2, x_3, \dots, x_n$  minus the value of the threshold, represented mathematically in eq.2.

$$G = \sum_{i=0}^n w_i * x_i = -T + \sum_{i=1}^n w_i * x_i, \quad w_0 = -T, \quad x_0 = 1 \quad (eq.2)$$

As the threshold value have already been used to calculate the value of  $G$  in a perceptron neuron, the threshold function used is no longer the same as the one used by the McCulloch and Pitts neurons. Instead, the threshold function used by perceptrons compare the value of  $G$  with 0 as in eq.3.

$$f(G) = \begin{cases} 1, & \text{if } G \geq 0 \\ 0, & \text{if } G < 0 \end{cases} \quad (eq.3)$$

All the weights in the first-generation artificial neural networks are changeable, thereby allowing the threshold value to be updated by the network itself, by changing the value of  $w_0$ . Fig.2 illustrates the perceptron model with the threshold input:

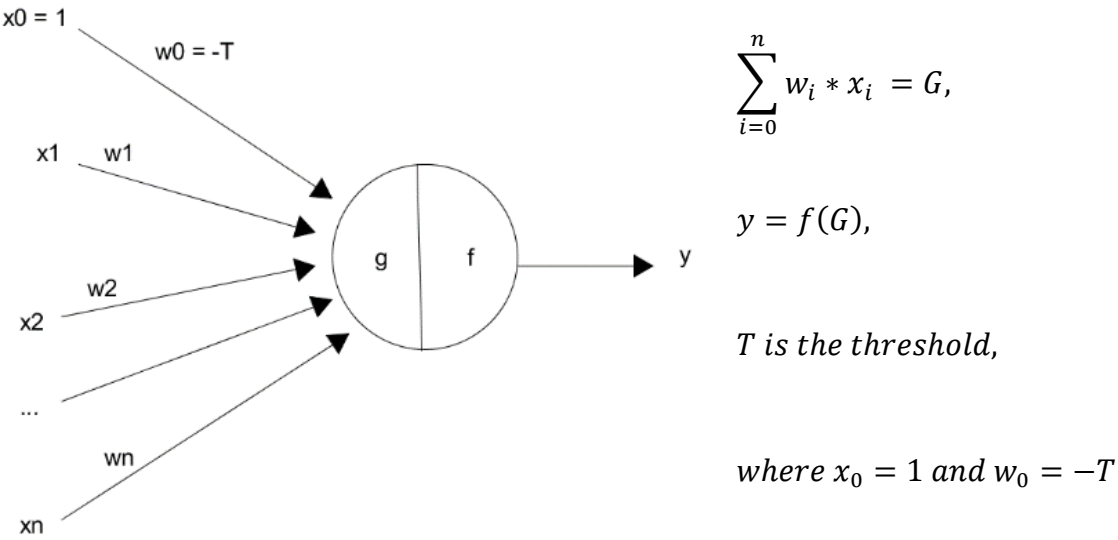


Fig.2: The perceptron neuron model.

The first-generation artificial neural network could then be trained by implementing some form of learning rule. The learning rule should tell the network how individual weights in the network should be updated, given some input data and the associated output. In addition to making the network able to learn, the introduction of weights meant that different importance could be given to inputs to a perceptron neuron. As a result, first-generation neural networks became significantly more biologically plausible compared to networks of McCulloch and Pitts neurons. The reason for why being able to assign different importance to inputs result in more biological plausibility, is easily demonstrated with an example:

A network of threshold perceptrons aim to make biologically plausible decisions on how to react, given some data received detailing a person. The firing of a single, specific neuron in the network  $N_0$ , represent the network making the decision to run away from the person. The input to  $N_0$  represent the following information of the person, being presented to the network:

$x_1 = \text{showing rage}, x_2 = \text{has a weapon}, x_3 = \text{is a newborn}, x_4 = \text{is frowning}$

It is safe to assume that a biological neural networks decision to run would be influenced in different degrees by if given the information in  $x_1$  through  $x_4$ . Most likely  $x_1$  and  $x_2$  would have a bigger impact on the decision, than  $x_4$ . Further, if  $x_3$  is true, the biological network might never decide to run away. The network of threshold perceptrons would be able to do the same, as each input  $x_1$  through  $x_4$  would have their own associated weights. Larger positive weights for  $x_1$  and  $x_2$  compared to  $x_4$ , would result in bigger input values for  $x_1$  and  $x_2$  than from  $x_4$ . As a threshold perceptron firing is based on the sum of all input values,  $G$ , bigger numbers for the input values will influence the value of  $G$  to a greater extent. Therefore, given some threshold value  $T$  and weights  $w_1, \dots, w_4$ , for the inputs, the neuron could be made to fire whenever  $x_1$  is true, or give  $x_1$  twice the amount of impact as  $x_4$ , all depending on the values of the weights. A weight being negative would result in the input being inhibitory for the firing of the neuron. By assigning an appropriately sized negative value for the weight  $w_3$ , the neuron  $N_0$  could be modelled so that the neuron would never fire, whenever the case for  $x_3$  is true ( $x_3 = 1$ ). By giving an input a weight value of 0, the said input becomes irrelevant for the firing of the receiving neuron.

A biological neural network deciding whether to run or is certainly more complex than being the result of a single neuron firing in the network. However, the example is sufficient in demonstrating that implementing weights in an artificial neural network result in the network being able to mimic biological neural networks to a greater extent. An increased ability to mimic biological neural networks indicate that the artificial neural network is more biologically plausible.

The decision concerning whether to run or not in a biological neural network is certainly not decided by the firing of a single. However, the example is still adequate to illustrate that being able to give different importance to the inputs result in the artificial neural network being able to mimic a biological neural network more closely.

An aspect of first-generation artificial neural networks that is not biologically plausible however, is the synchronization of neural computation in a layer of neurons. Neurons in



first-generation artificial neural networks compute their output for each arbitrary discrete time step for  $t = 0, 1, 2, 3, \dots$  [31]. The reason for the synchronization is sum operation used by perceptron neurons to decide its own output. Therefore, every neuron in the first layer of the network will calculate their values before sending their outputs at  $t = 0$ . The outputs from the neurons in the first layer will therefore be received synchronously by the neurons in the second layer. The neurons in the second layer will all send their outputs to the next layer at  $t = 1$ , and so forth. Referring back to chapter 2.1, there is no reason as to why a biological neuron should be able to receive one signal at a given time and another half a millisecond later. Furthermore, calculations performed by conventional computers are done in a sequential manner, albeit extremely fast, resulting in some of the outputs of neurons in the same layer being finished computing ahead of some other. Therefore, the necessity of synchronizing the inputs to a layer can result in neurons being finished computing their output.

The learning methods used in 1<sup>st</sup> generation neural networks, was most often based on Hebb's rule, stating: "Cells that wire together, wire together", or more accurately "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased" [10]. It is believed that biological neural networks also implement some forms of Hebb's rule.

Hebb's rule can be written mathematically as in eq.4, using the same notations as in fig.2:

$$\Delta w_i = n * yx_i \quad (eq. 4)$$

Where the variable  $n$  in eq.4 is a defined learning rate. A specialized case of Hebb's rule called Oja's rule, is the one usually being implemented in artificial neural networks for learning. Oja's rule adds a term to eq.4 so that if cell B fire without A, the weight of the synapse between them is weakened. Oja's rule can be written mathematically as in eq.5:

$$\Delta w_i = n * yx_i - n * y * w_i * y = ny(x_i - w_i * y) \quad (eq. 5)$$

For two boolean perceptrons connected by a weight, perceptron B firing would mean  $y = 1$ , and perceptron A firing would mean  $x_a = 1$  in eq.5. Note that in Hebb's rule, there is no term for a "target value", and as such Hebbian Learning algorithms are unsupervised in nature. To achieve supervised learning, one would need to add the target term, the defined correct output for a given input, to the equation. The perceptron learning rule [19] does this by adding the target value as shown in eq.6:

$$\Delta w_i = n(t - y)x_i \quad (eq. 6)$$

Where  $t$  in eq.6 is the target value for the output of the neuron, given the inputs. The perceptron learning rule have clear similarities to Hebb's rule evident by comparing eq.4

with eq.6. An alternative to the perceptron rule for a supervised learning method, which is based on a similar principle to Hebb's rule, is the correlation learning rule. The only difference between the two is that the correlation learning rule replace the output variable,  $y$  in eq.4, with the variable for the target value,  $t$ , as in eq.7:

$$\Delta w_i = n * tx_i \quad (eq.7)$$

Supervised learning methods have existed as long as artificial neural networks. However, first generation artificial neural networks were unable to implement supervised learning in networks consisting of one or more hidden layers. The reason being supervised learning require some optimization technique to be implemented. For an artificial neural network to use optimization techniques available from calculus, the neuron outputs are required to be continuous. However, outputs from perceptrons in first-generation artificial neural networks are discontinuous, as the threshold function used, gave the output as a boolean value. First-generation artificial neural networks lacked therefore implementable optimization techniques. As such, unsupervised learning methods based on Hebb's rule were the learning methods that saw the most use by first generation artificial neural networks.

### 2.2.2. Second Generation of Artificial Neural Networks

The second-generation artificial neural networks would use the same perceptrons (fig.2) as the first-generation neural networks. The difference between the two generations as defined by Maass, is the function,  $f(G)$ , used to calculate the output values for the perceptrons. For second-generation artificial neural networks, the perceptrons no longer used a threshold function. Instead, an activation function is used, mapping the input to an output  $y \in [0, 1]$ . The use of an activation function meant that the output of perceptrons in second-generation artificial neural networks became a continuous real value. Several activation functions exist, such as sigmoidal functions. An example of a sigmoidal function often used by perceptrons as an activation function, is the logistic function, given by eq.8:

$$f(G) = \frac{1}{1+e^{-G}} \quad (eq.8)$$

As the output value changed to real values, it was no longer possible to view the output as single spikes. Hence it became necessary to find a new biological interpretation of the output from perceptron neurons in second-generation neural networks. The biological interpretation became that of the "current firing rate" of the neuron [55]. Instead of giving an output, 1 or 0, true or false, as the first generation neural network, the output from a perceptron using an activation function, would give its answer as a probability.

Second-generation neural networks would still synchronise the neuron output for each layer, the same way as first-generation neural networks. The only difference between the

two generations concerning time is the output of neurons in second-generation neural networks being seen as the firing rate of the neuron. As a firing rate would be achieved by averaging the frequency of spike-generation over a time window, it can be argued that second-generation artificial neural networks implement a notion of time implicitly.

As the output of perceptrons in the second-generation artificial neural networks was no longer single spikes, the networks became unable to directly implement Hebbian Learning. Nevertheless, the fact that the output from the perceptrons changed from boolean to real values, allowed for the implementation of optimization techniques present in calculus based on gradient descent. Therefore, the most used learning method changed from unsupervised Hebbian learning to supervised learning algorithms based on gradient descent optimization techniques.

As of today, nearly all the artificial neural networks used, belong to the second-generation of artificial neural networks. The most used learning algorithm deployed by second-generation artificial neural networks, is the supervised learning method called Backpropagation. However, the longevity, of backpropagation has come into question in recent years. Geoffrey Hinton, a pioneer in deep learning, credited for popularizing backpropagation by co-publishing “Learning Internal Representations by Error Propagation” [44] in 1986, made the following statement concerning backpropagation in 2017: “My view is throw it all away and start again” [28]. The biological plausibility of supervised learning methods was also questioned by Hinton. His view was to shift the focus of research, from supervised to unsupervised learning methods. As the main benefit of second-generation artificial neural networks is their ability to implement supervised learning methods, it could be worth considering looking for other alternatives.

### 2.3. Third Generation of Artificial Neural Networks

Third-generation artificial neural networks are defined as artificial neural networks, implementing spiking neuron models as computational units [55]. While the transition from discontinuous spike-outputs in first-generation neural networks, to continuous fire-rate outputs in second-generation networks lead to a big leap in training opportunities, spiking neuron models are back to outputting single spikes. The reversion back to discontinuous output, might seem counterproductive as the ability to perform optimization techniques, such as gradient descent, directly is thereby lost by spiking neural networks. Both the first- and third-generation artificial neural networks give their outputs as neuron spikes.

The difference to the earlier generation, is that spiking neuron models implements the notion of time explicitly, in the form of the change in membrane potential of the neuron over time. Therefore, both neural inputs and outputs gain a new dimension, the temporal axis. The importance of the implementation of time into the model will be elaborated in later chapters. As spiking neural networks implements time explicitly the outputs of spiking

neuron models become spike-trains (a timeline showing spike-formation(s) by the spiking neuron at distinct time points). Since spiking neural models no longer output a single number, such as the perceptron model, it became necessary to find methods for interpreting the spike trains, as the network has to be able to form decisions based on its output. Therefore, chapter 3 is dedicated to discussing some of the common ways to interpret spike-trains in spiking neural networks. There are some that argue that not all neural networks implementing spiking neurons should be classified as spiking neural networks, and that it depends on how one chose to interpret the output signals. However, for the rest of this survey, all artificial neural networks consisting of spiking neuron models will be considered as spiking neural networks, no matter the method chosen for spike-train interpretation.

### 2.3.1. Spiking Neuron Models

Spiking neuron models is not a new concept by any means. One of the first models, the integrate-and-fire model can be traced [52] back to the work of Lapicque (1907) [6]. However, Lapicque himself did not define the model, which was defined a while later. As such, many consider the Hodgkin-Huxley model (1952) to be the first spiking neuron model.

As of today, many different spiking neuron models have been proposed. The names of the most well-known spiking neuron models are included in fig.3 below:

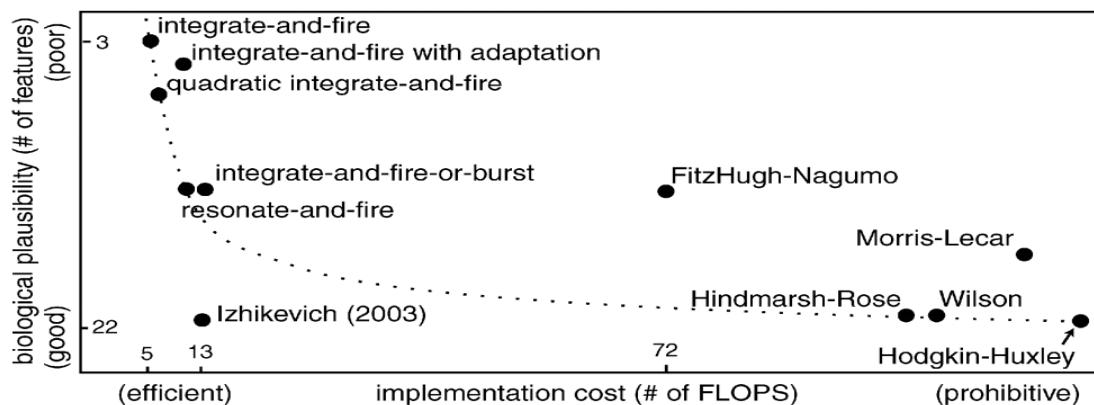


Fig.3: Graph comparing spiking neuron models based on biological plausibility and computational cost (figure taken from [13]).

““# of FLOPS” is an approximate number of floating point operations (addition, multiplication, ...) needed to simulate the model during a 1 ms time span.” The Hodgkin-Huxley model have an implementation cost of 1200 flops, while the integrate and fire model have an implementation cost of 5 flops [13].

It is possible to divide spiking neuron models into one of two modelling directions in computational neuroscience, detailed neuron models and simplified neuron models [9]. The Hodgkin-Huxley model, maybe the most well-known example of a detailed neuron model, strives to be as biologically plausible as possible. The integrate and fire model, is an example

of a simplified neuron model, trying to model the spiking nature of the neuron, while remaining as computationally inexpensive as possible. The optimal model to use for a given spiking neural network depends on the desired results for the network question. For mimicking biological neurons as closely as possible, the Hodgkin-Huxley model might be optimal. If the desire is to create a large spiking neural network however, the Hodgkin-Huxley model become too computationally expensive. Therefore, for larger spiking neural networks simplified neuron models, such as the leaky integrate and fire model, should be used, as to limit the necessary computational operations for simulating the networks.

The survey will introduce both the Hodgkin-Huxley model, the (leaky) Integrate and fire model, as well as a newer Izhikevich model which tries to combine the biological plausibility of the Hodgkin-Huxley model with the computational efficiency of the integrate and fire model.

### 2.3.1.1. The Hodgkin-Huxley Model

The Hodgkin-Huxley neuron model was proposed by Hodgkin and Huxley in their paper “A quantitative description of membrane current and its application to conduction and excitation in nerve” [1]. In their paper, Hodgkin and Huxley proposed a mathematical model for the creation of an action potential based on an input current,  $I_{input}$ , and the sum of ionic currents passing through their respective ion channels of the cell membrane. In their original work, they included potassium ( $K$ ) and sodium ( $Na$ ) as well as a leak ( $L$ ) as ionic currents.

The Hodgkin-Huxley model represent the cell membrane of the neuron with a capacitor,  $C_m$ . The membrane potential at a given time  $t$  is the voltage  $V_m$  associated with the capacitor. The relationship between the membrane potential and the cell membrane is described by eq.8:

$$C_m = \frac{q}{V_m(t)} \quad (eq.8)$$

The equation allowing for simulating the Hodgkin-Huxley neuron (eq.9) calculate the change in membrane potential, and the capacitance of the capacitor at a given time  $t$ , from an input current and the flow of ionic currents passing through the cell membrane.

$$C_m \frac{dV_m(t)}{dt} = I_{input}(t) - \bar{g}_K n^4 (V_m(t) - E_K) - \bar{g}_{Na} m^3 h (V_m(t) - E_{Na}) - \bar{g}_L (V_m(t) - E_L) \quad (eq.9)$$

By using the relationship between  $C_m$  and  $V_m$ , as given in eq.8, it becomes possible to calculate the current membrane potential at any given time from eq.9. A spike is then generated if the value of the membrane potential reach a given threshold.

To explain how the Hodgkin-Huxley model allow for the simulation of neuron spike-generation, the parameters in eq.9, as well as how the parameters change the membrane potential, will be given. The aim of the explanation is to demonstrate the computational complexity of the Hodgkin-Huxley.

The variables  $n$ ,  $m$  and  $h$  in eq.9 are gained from a set of 3 other differential equations. The differential equations modelling  $n$ ,  $m$  and  $h$  are not included in this survey, as it is sufficient to know that all the variables depend on the value of  $V_m(t)$ . The rest of the variables  $\bar{g}_{ion/L}$ , and  $E_{ion/L}$ , are all constants associated with said ion/leak.<sup>1</sup>

Hence to calculate if a Hodgkin-Huxley neuron will generate a spike from an input current, first one need to solve the differential equations for variables  $n$ ,  $m$  and  $h$ , before using said variables in eq.9. The variables  $n$  and  $m$  increase with  $V_m$ , while  $h$  decrease (shown in fig. 2.6c in [50]). Further, sodium ions flow into the cell, increasing the membrane potential, while potassium ions flow outwards, decreasing the membrane potential. The potassium ion flow reacts slower to change in the membrane potential, compared to the sodium flow however (shown in fig. 18 in [1]).

The flow of sodium ions is increased by an increase in  $m$ , while the flow decreases as  $h$  decrease. However,  $h$  reacts slower than  $m$  to a change in the membrane potential. Therefore, since  $m$  increase with an increase in  $V_m$ , an increase in the membrane potential results in more sodium ions flowing into the cell, which in turn, further increase  $V_m$ . As such, a positive feedback loop is initiated where  $V_m$  and  $m$  will continue increasing in unison. Depending on the values of the membrane potential, and the input current, the positive feedback loop generated by  $V_m$  and  $m$  might result in the membrane potential exponentially increasing, thereby resulting in the generation of a spike [16]. The feedback loop will continue until either a neuron spike is generated, or the membrane potential start to decrease as the change in  $h$  start to limit the sodium flow, as well as the potassium flow start to change, decrease the membrane potential directly.

When a spike is generated, the exponential increase in the membrane potential from the positive feedback loop, will come to a natural halt when  $V_m$  approach  $E_{Na}$ , resulting in no more sodium ions flowing into the cell. The spike generation is followed by a sharp decrease in the membrane potential towards its resting value.

Although the Hodgkin-Huxley model is the most biological plausible spiking neuron model, it is simply far to computationally expensive, due to the feedback loop initiated, requiring eq.9 and the differential equations for  $n$ ,  $m$  and  $h$  to be calculated continuously, in order to see if a spike is generated. The Hodgkin-Huxley model is therefore not a realistic implementation choice for any large-scale spiking neural network.

Nevertheless, the extensive computation does have the benefit of making the Hodgkin-Huxley model one of the most biologically plausible spiking neuron models available. As

---

<sup>1</sup>  $\bar{g}_{ion/L}$  is an approximated constant based on the ionic conductivity, while  $E_{ion/L}$  is the Nernst potential of the ion/ leak in question.

such, the Hodgkin-Huxley model can replicate a wide variety of spike shapes and patterns (shown in fig.2 in [13]).

### 2.3.1.2. The Integrate and Fire Model

The integrate and fire neuron model is far less computationally complex than the Hodgkin-Huxley model. Therefore, a variation of the integrate and fire model, the leaky integrate and fire model, is the most implemented spiking neuron model in spiking neural networks so far [50]. The leaky integrate and fire model can be explained mathematically by eq.10 [7]:

$$C_m \frac{dV(t)}{dt} = I_{input}(t) - \frac{C_m}{\tau_m} (V(t) - V_0) \quad (eq. 10)$$

In eq.10,  $V(t)$  represent the membrane potential at time  $t$ , while  $V_0$  is the resting potential of the membrane. The variable  $\tau_m$  is equal to the resistance of the membrane,  $R_m$ , multiplied with the capacitance,  $C_m$ . The variables  $R_m$  and  $C_m$  are both assumed to be constant in the leaky integrate and fire model. As such, the equation can be rewritten:

$$V(t) = R_m I_{input}(t) + V_0 - \tau_m \frac{dV(t)}{dt} = a * I_{input}(t) - b * \frac{dV(t)}{dt} + c \quad (eq. 11)$$

Where  $a$ ,  $b$  and  $c$  in eq.11 are all constants. If  $V(t)$  in eq.11 reach a certain threshold, a spike is generated and  $V(t)$  is reset to equal  $V_0$  (often set to 0 as a common assumption [40]). The term  $-b * \frac{dV(t)}{dt}$  in eq.11 represent the cell membrane “leaking” ions, changing the membrane potential towards its resting value over time (hence the name leaky integrate and fire).

Similarities between eq.9 and eq.11 can be drawn. Both the Hodgkin-Huxley model and the leaky integrate and fire model, use a capacitor for modelling the cell membrane. Both eq.9 and eq.11 can be written as eq.12:

$$C_m \frac{dV_m(t)}{dt} = I_{input}(t) - something \quad (eq. 12)$$

Although there are some similarities between the two spiking neuron models, there are more differences. As the leaky integrate and fire model does not require the computation of an extra three differential equations, and no feedback loop initiated, the model becomes significantly less computationally heavy compared to the Hodgkin-Huxley model. The reason as to why the leaky integrate and fire model does not need extra differential equations, is due to the model assuming the shape of the action potentials is uniform for all spikes [40]. However, assuming that the shape of all spikes is uniform severely limits the ability to

reproduce observed spiking patterns (and of course spike-shapes) thereby, decreasing the biological plausibility of the model. Nevertheless, the assumption of spike-shape uniformity makes it possible for creating large spiking neural networks, as it is possible to simulate hundreds of thousands of leaky integrate and fire neurons[50].

### 2.3.1.3. The Izhikevich Model

The Izhikevich model was proposed by Izhikevich in 2003 [12], which sought to be both computationally simple and biologically plausible (see fig. 3). The Izhikevich model is able to reproduce all known spiking patterns as well as spike shapes of biological cortical neurons (shown in fig.2 in [13]), while being close to the integrate and fire model in computational complexity (the integrate and fire model require 5 flops for simulating the model over a 1ms time interval, while the Izhikevich model require 13 flops).

The Izhikevich model is described by a combination of three equations, eq.12, eq.13 and eq.14:

$$\frac{dv(t)}{dt} = 0.04v^2 + 5v + 140 - u + I_{input}(t) \quad (eq. 12)$$

$$\frac{du(t)}{dt} = a(bv - u) \quad (eq. 13)$$

$$if \ v \geq 30mV, then \ \begin{cases} v \leftarrow c \\ u \leftarrow u+d \end{cases} \quad (eq. 14)$$

Where  $v$  is the membrane potential at a given time, and  $u$  is the “membrane recovery variable”, a variable that decide how fast the membrane potential goes towards its resting potential, given the current membrane potential.  $a, b, c$  and  $d$  are constants set depending on what type of neuron wished to model. A spike is generated when  $v$  in eq.12 reach a certain threshold, then when the membrane potential reach the value of the “spike apex” (30mV) both  $v$  and  $u$  are reset according to eq.14. According to Izhikevich, they used the model to simulate 10 000 spiking neurons with 1 000 000 synapse-connections in real time (resolution of 1 ms) using a 1GHz desktop PC.

## 2.3.2. Spiking Neural Networks: A Network of Spiking Neurons

Although spiking neuron models have different degrees of biological plausibility, they are all notably more biologically plausible than perceptron models from both the first and second generation artificial neural networks. Spiking neural networks are fundamentally different compared to earlier generations of artificial neural networks as a result of the increased biological plausibility. The biggest difference would be the inclusion of the temporal axis in both the input and output. The addition of the temporal axis allows spiking neural networks



to update their neurons in an asynchronous fashion, allowing faster response times of the output signals compared to other artificial neural networks. However, the addition of the temporal axis, although a significant step towards biological plausibility and directly beneficial in certain applications, bring with it several new challenges that spiking neural networks must overcome. The rest of the survey is aimed at explaining different aspects of spiking neural networks, revolving around the benefits and challenges associated with the newly added notion of time to the networks. Chapter 3 concerns how information is processed in spiking neural networks, as spike-train outputs generated over time is inherently different than a single number generated as the output for a given arbitrary timestep. Chapter 4 deals with how to implement both supervised and unsupervised learning methods in spiking neural networks. Lastly chapter 5 and 6 summarises some of the specific benefits and challenges of spiking neural networks, respectively. The future of spiking neural networks is discussed in the last chapter, chapter 7, as a conclusion.

### 3. Neural Coding

In second-generation neural networks the output layer would give a probability value for a given outcome. If a second-generation neural network was presented with a still image of a written number, the output layer of the network would contain a neuron for every possible solution to the image. Every neuron in the output layer would calculate an output, representing the probability for each possible solution. The solution associated with the highest output value from the output layer would then be given as the answer by the network. Preferably, if trained correctly, the probability of one outcome should be clearly higher than the rest.

However, the situation becomes more complicated with spiking neural networks. In spiking neural networks, the output neurons will output sets of spikes over time, not a single probability value. Therefore, neural coding, a topic in the field of Neuroscience, becomes relevant when discussing spiking neural networks. Neural coding turns out to be an essential topic for spiking neural networks, as a neural network will first become useful when it is possible to interpret meaningful data from the network. Due to the biological plausibility of spiking neuron models, theories of how biological neural networks encode information in the form of spike events over time, become the basis for how to interpret spiking neural network outputs. However, the way biological neurons encode and decode spike information is still an unanswered question in the Neuroscience community, commonly referred to as the question of the neural code [5], [18].

Currently two main theories as to how biological neurons encode information exist: The first, which was the main theory for a long time, is the idea that information is encoded in the form of the average spike frequency in a spike train. The second theory revolves around the idea that information is encoded in the specific timing of individual spikes in spike trains – the temporal information in spike trains. While spiking neural networks should in theory have a constant output in time, the machine used to calculate the output have to deal with discrete time values, therefore, a time-resolution is associated with spiking neural networks. Due to the need for a time-resolution, a timestep is also present in spiking neural networks. The timestep is equal to the time resolution of the spiking neural network. The computer simulating the network will not make any computations for a time  $t$  fulfilling the conditions of eq.15.

$$t_{prev\ calculated} < t < t_{prev\ calculated} + time\ resolution \quad (eq.15)$$

#### 3.1. Rate Coding

Rate coding is the most implemented neural coding scheme in spiking neural networks [4]. Rate coding find the average spiking frequency of a neuron over a certain timeframe and use the calculated rate mostly in the same manner as output from second-generation neural networks are used. Rate coding can be applied to single spiking neurons and is easy to implement. However, by dealing with rates, no temporal information is gained, which is one of the biggest advantages spiking neuron models have over earlier neuron models. It is also worth to note that, due to the nature of finding rates, which require a certain time window, rate coding is inefficient compared to temporal coding, both when it comes to information capacity and transmission speed. Using rate coding instead of a form of temporal coding, leads to higher latency and energy consumption in deep spiking neural networks [38]. The introduced latency is due to the needed time to calculate an average firing rate. For transmitting 4 bits of data, a spiking neuron implementing rate coding would have to use  $2^4 = 16$  timesteps. The time needed for a spiking neuron to transmit bits exponentially increase with the number of bitts to be transmitted. While transmitting 4 bits took 16 timesteps, transmitting 8 bits of data would require  $2^8 = 256$  timesteps. The rate coding theory which can be traced back to 1926, was the dominant theory for how the brain encode information in spikes for a long time. However newer research suggest that rate coding cannot account for everything about how information is processed in the brain [9], [53].

### 3.2. Spike Coding

Spike coding is a set of many different temporal coding schemes. All temporal coding schemes suggests the information of the neuron output is encoded in the temporal information of individual spikes. Three of the most common spike-coding schemes will be mentioned, namely: time to first spike (A), rank order coding (B), and latency coding (C). A visualization of the three temporal coding schemes is added as fig.4:

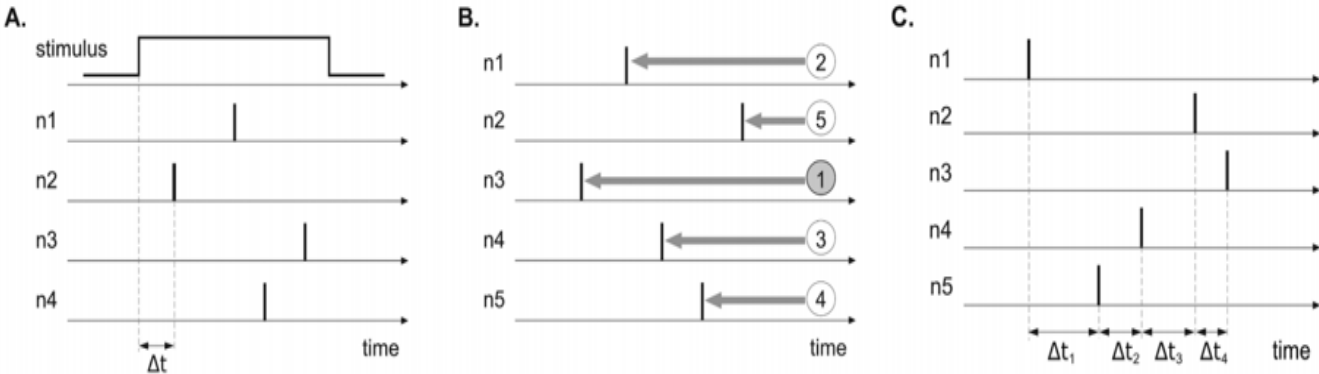


Fig.4: Visualizing different temporal coding schemes and how they approach information encoding in spike trains. (figure taken from [42])

#### A) Time to first spike:

Time to first spike (TTFS) neural encoding assumes, as the name imply, that the information present in the spike trains is encoded by the time difference between stimulus onset to the first neuron spike ( $\Delta t$  in fig.4A). A further assumption in TTFS encoding is that a neuron generates only one spike for any given stimulus, therefore if subsequent spikes follows from said neuron, they are simply ignored [36]. TTFS has been implemented by several spiking neural networks, and can be implemented using just one single neuron. An advantage of TTFS is that it allows for very fast information processing, as the information is already transmitted when the first spike in the spike train(s) form from a stimulus. An example of a spiking neural network that use TTFS encoding is given in [39]: “T2FSNN: Deep Spiking Neural Networks with Time-to-first-spike Coding”. [39] achieved 99.33% on MNIST, using a deep neural network to spiking neural network (DNN-to-SNN) conversion learning method (which will be explained in chapter 3). [39] also showed good results in both latency and number of spikes generated compared to other neural coding schemes using DNN-to-SNN.

#### B) Rank order coding:

Rank order coding (ROC) cannot be implemented by a single neuron but must be used on a population of neurons. The idea behind it is, nevertheless, relatively simple. The exact timing of each spike is not considered, instead only the order of firings from a set of neurons is what encodes the information in ROC. From Fig.4B the order would be  $n_3 > n_1 > n_4 > n_5 > n_2$ . ROC encoding assumes that each neuron only fires once the same way TTFS does. However even if every neuron only fires once, in the case of  $N$  neurons, there will be  $N!$  ( $N$  factorial) possible orderings. Therefore, spiking neural networks implementing ROC can transmit up to  $\log_2(N!)$  bits of information from a population of  $N$  neurons, all firing just once [51]. [29] is written by the one that proposed the ROC method, detailing a spiking neural network, implementing ROC encoding for speech recognition.

#### C) Latency coding:

Latency coding is the coding scheme that can contain the most information based on the number of spikes, from the three mentioned spike coding schemes. The idea behind latency coding is that the information is encoded in the relative timing between each output spike by a set of neurons ( $\Delta t_1, \Delta t_2, \Delta t_3$  and  $\Delta t_4$  in fig.4C). Latency coding assumes

only one spike for each neuron just as TTFS and ROC does. A temporal coding scheme often just referred to as temporal coding, is an extension of the idea of latency coding. Temporal coding use the relative spike times, the same of latency coding, however, further include the time from stimulus-onset to the first spike, as an extra piece of information. To the authors knowledge, temporal coding is the coding scheme that store the most information per spike generated of all spike coding schemes. Latency coding was used in the paper [45] that proposed the supervised learning algorithm SpikeProp for spiking neural networks.

In a biological neural network, it is likely that both rate coding together with one, or several temporal coding schemes is used in combination, as to store and transmit the most amount of data possible. However, trying to implement several coding schemes in artificial spiking neural networks, result in learning methods, especially supervised learning methods becoming increasingly hard to implement. It is therefore not usual to implement more than one temporal coding scheme. However, several instances for learning methods in spiking neural networks, using rate coding together with one single temporal coding scheme, exist.

## 4. Learning in Spiking Neural Networks

Chapter 1 and 2 dealt primarily with how one can set up a spiking neural network, from the elementary building blocks, the neuron models used, as well as different ways to interpret the output. However, one last thing remains for spiking neural networks to be of much use, which is the ability to learn. Learning in spiking neural networks is also based on the notion of changing synaptic weights, as to facilitate strengthening or weakening of spiking correlation between neurons. The learning process is complicated in spiking neural networks compared to earlier-generation neural networks, since spiking neural networks can be implemented with many kinds of both neuron models as well as neural coding schemes. Therefore, learning methods often have to be specifically created to suit the specific type of spiking neural network, regarding the neuron model and coding scheme used by the network. There is reason to believe that spiking neural networks would benefit from the same learning mechanisms as biological neural networks, due to their inherent spike-based similarities. However, learning mechanisms deployed by our brains is still, not understood in much detail. Most likely, different areas of the brain deploy different kind of learning mechanisms, and the learning mechanism used might change during a lifetime [25].

This chapter will focus on supervised learning as well as unsupervised learning. Further, a method of learning, by converting a deep neural network to spiking neural network (DNN-to-SNN) will be mentioned. Lastly some recent performance results by spiking neural networks will be compared to the state-of-the-art results achieved by sigmoidal neural networks. Accuracy results on the MNIST benchmark will be used as a basis for comparison.

### 4.1. Supervised learning

Supervised learning methods for training are limited to situations where both input data, and labelled answers of the inputs are available. Supervised learning methods feed data to an artificial network, and give feedback on the performance of the network, by comparing the actual output of the network, with the correct answers. In order for comparing the output of spiking neural networks with correct answers, it is necessary to both convert the given input data to spike-trains, as well as finding a way to compare the output spike train to the correct answer. Several ways for comparing spike trains with the labelled data have been proposed. Some, convert the labelled data to spike trains, and give feedback based on the difference in spike timing, others simply try to use the firing rate of spikes in the spike train to gain a number value for comparison. All the methods have their own associated challenges, making it harder for supervised learning methods for spiking neural networks to give accurate feedback to the network. Further, as spiking neural networks are back to

outputting discontinuous data, spiking neural networks cannot implement gradient descent without doing some extra work.

The Hodgkin-Huxley model can give out a continuous output, in the form of the change in membrane potential with time, thereby solving the discontinuous output problem of spiking neural networks. However, the model is too computationally complex to consider using for any large-scale spiking neural network, as stated before.

Solutions like Supervised Hebbian Learning and an improved version named ReSuMe, have been successful in creating a form of supervised learning for spiking neural networks. However, they are mostly limited to single-layer networks [42]. Hence, their implementation potential becomes limited, due to artificial neural networks mostly consisting of several layers, and the increasing importance of deep learning. Several other supervised learning methods have been developed since based on the same principles as ReSuMe and Supervised Hebbian Learning, while outperforming both. [30] is such an example, however, the newer developed methods, still suffer from the same limitations as ReSuMe and Supervised Hebbian Learning.

Other solutions, such as SpikeProp, are supervised learning methods that modify backpropagation, as to make the method implementable by spiking neural networks. SpikeProp works by minimizing the difference between target spike-time and output-spike time using gradient descent. The method was first limited to single spike-emission, but has seen extensive work since its creation, being adapted to allow for both faster learning convergence and multiple spike emissions [49]. However, SpikeProp place limitations on both the neuron model, as well as the neural encoding scheme of the spiking neural networks implementing SpikeProp. Many modified forms of backpropagation have been proposed since SpikeProp, however all modified backpropagation methods place some limitations on the neural network, concerning the neuron model, or the neural encoding scheme used. Nevertheless, modified backpropagation solutions for spiking neural models have seen a steady improvement of accuracy performances the last years, which can be seen in the table for MNIST result in chapter 3.4. The earlier versions of modified backpropagation models for spiking neural networks assumed that rate-coding was used. However, in recent times, spike-based backpropagation solutions have increasingly become the focus for modified backpropagation methods proposed. All modified backpropagation learning methods create approximations of the output, as to make them differentiable. Discontinuities in the output spike-trains are often simply seen as noise, simply being ignored by the method.

In “Spiking Neural Networks: Principles and Challenges” [18] it was stated that no general-purpose (supervised) learning algorithm exist for spiking neural networks. The author has failed to find any newer publications that go against this statement.

## 4.2. Unsupervised Learning

Most forms of unsupervised learning in spiking neural networks is based on Hebbian learning. The specialized case of Hebbian learning specifically adapted for use by spiking neurons, is Spike-time-dependent-plasticity (STDP), sometimes referred to as the temporal Hebbian rule. STDP is based on the same principle of Hebbian learning – “neurons that fire together wire together”, the difference between STDP and Hebbian learning, is the need to define a timeframe for when neurons are said to “fire together” in STDP. STDP, as a mechanism for synaptic strength modification, has strong biological support from detailed experiments, such as [3]. In [11], they modelled a best-fit line for synaptic modification. The best-fit line was based on a set of single spike outputs from experimental data, of two connected neurons. Their results were a curve similar to the illustration provided in fig.5:

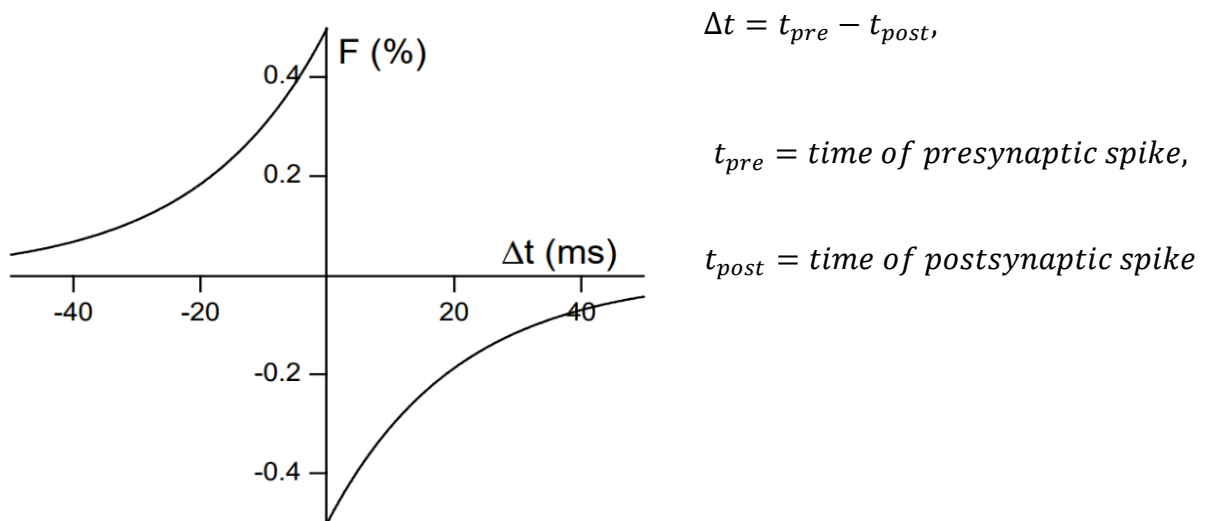


Fig. 5: Illustrating synaptic weight modification from STDP in percentage. (figure taken from [47])

The y values of the graph in fig.5 will naturally vary, depending on the learning rate and defined time interval range chosen. The time interval range is the time interval that allows synaptic modification to occur. The focus for studying fig.5 should be the curve shape, illustrating the increasing change in weight, as the time difference between the two spikes  $\Delta t$  approach zero. When the time difference  $\Delta t$  grows, the weight change will decrease.



The equation of weight change by STDP from a single spike event in two connected neurons, can then take the form of eq.16.

$$\Delta w = \begin{cases} Ae^{-\frac{|\Delta t|}{T}}, & \Delta t \leq 0, A > 0 \\ Be^{-\frac{|\Delta t|}{T}}, & \Delta t > 0, B < 0 \end{cases} \quad (\text{eq.16})$$

The variable  $T$  in eq. 16 is the defined timeframe for which synaptic modification should occur, while  $A$  and  $B$  are constants indicating the learning rate. Eq.16 is found in both [47], [49], however, spiking neural networks rarely deploy the exact form of STDP given in eq.16. Instead, variations of eq.16 is used depending on the system needs [49]. [24] is an example of STDP learning being deployed in a similar manner to eq.16. A deep spiking-CNN is used for object recognition, achieving 98.4% accuracy on MNIST with the given STDP method.

A STDP model for spike trains, as compared to being limited to of single spike outputs for each neuron as eq.16, is given in eq.17. The STDP model for spike trains was proposed in 2002 by Gerstner and Kistler [15]:

$$\frac{d}{dt} w_{ji}(t) = a_0 + a_1 S_i(t) + a_2 S_j(t) + a_3 S_i(t) \bar{S}_j(t) + a_4 \bar{S}_i(t) S_j(t) \quad (\text{eq.17})$$

In eq.17,  $w_{ji}$  is the synaptic weight between neuron  $i$  and  $j$ .  $S_i$ ,  $S_j$  are the spike train outputs by the two neurons. The spikes in the spike trains are modelled as instantaneous pulses in time (illustrated in fig.6).  $\bar{S}_i$ ,  $\bar{S}_j$  are lowpass filters of their associated spike trains (also referred to as the “synaptic trace” – exponential decay of the spikes in the spike trains, instead of instantaneous as to retain a “spike history”).  $a_i$  are constants controlling the rate of change in  $w_{ji}$ , where  $a_0$  specifically controls the weight decay over time [42]. Eq.17 is best explained accompanied with an illustration, fig.6:

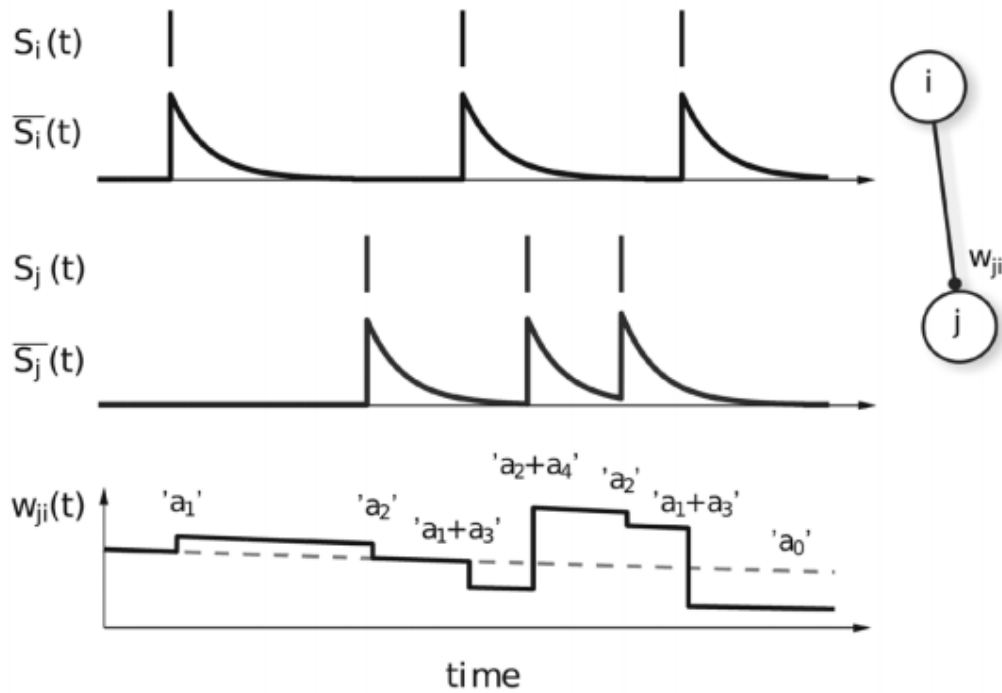


Fig.6: Illustrating STDP model given in eq.17. The figure shows how the synaptic weight,  $w_{ji}$ , between two neurons  $i$  and  $j$ . The change is given by a set of values for  $a_i$ , the spike trains  $S_i$ ,  $S_j$  and their synaptic trace  $\bar{S}_i$ ,  $\bar{S}_j$ . (figure taken from [42])

### 4.3.DNN-to-SNN conversion

DNN-to-SNN conversion is a way to train spiking neural networks by converting an already trained traditional deep neural network, trained with backpropagation, to a spiking neural network. The spiking neural network aim to retain the trained weights from the deep neural network, while introducing the necessary parameters for spiking neurons [41]. The neural encoding scheme used by the spiking neural networks are limited by the DNN-to-SNN conversion method. Most DNN-to-SNN conversion methods limit spiking neural networks to use rate encoding. The use of rate coding makes the transformation of the deep neural network to an spiking neural network simpler, as the analog output of the deep spiking neural network, can be directly recreated by the rate coding of spike train outputs in the spiking neural network. Hence, although some of the best accuracy results by spiking neural networks are from using DNN-to-SNN, much of the benefits of spiking neural networks are lost using the conversion method, as the network does not retain any temporal information, given that rate coding was used. However, some DNN-to-SNN methods, like [61], convert deep neural networks to temporal-coding spiking neural networks. The trend of using temporal-coding in DNN-to-SNN conversion learning methods, are growing.

#### 4.4. Performance of Spiking Neural Networks

Spiking neural networks have lagged behind traditional artificial neural networks in accuracy performances on datasets such as MNIST, both historically and till this day. However, the gap between the accuracy results achieved by spiking neural networks and traditional neural networks is decreasing steadily. To the authors knowledge, the current best result on the MNSIT dataset is an accuracy of 99.84%, achieved by a traditional neural network[8]. The best spiking neural network result found was 99.62% in comparison, included in table 1. The author could not find any unsupervised learning methods for spiking neural networks able to produce similar results, although spiking neural networks are well suited for unsupervised learning methods. The best results achieved from unsupervised learning on the MNIST dataset found by the author was 98.40% by [24], followed by [37] with an accuracy of 96.80% (as average accuracies).

DNN-to-SNN conversion methods are the learning methods that have achieved the highest accuracy results for spiking neural networks previously, however, direct learning on spiking convolutional neural networks have surpassed their conversion learning method counterparts. The fact that direct training of spiking neural networks has surpassed the conversion methods in some cases, bodes well for spiking neural networks. Although conversion methods have seen good improvements concerning minimizing accuracy loss, in the conversion between neural network types, it will not be possible for a spiking neural network to outperform a traditional artificial neural network by using conversion methods.

The reasoning behind which model has been included or omitted in table 1, is to show the best accuracy achieved during a given year, while illustrating the gradual improvement in accuracy seen for spiking neural networks. For a more extensive table over spiking neural models up to and including 2017, the survey will simply refer to [49]. [32] includes a smaller table over unsupervised learning method results on the MNIST dataset.

Model:	Architecture	Learning method	Year	Accuracy
Diehl [35]	spiking CNN	conversion	2015	99.1%
Lee [21]	spiking CNN	modified backprop	2016	99.31%
Rueckauer [2]	spiking CNN	conversion	2017	99.44%
Wu [58]	spiking CNN	modified backprop	2017	99.42%
Jin [57]	spiking CNN	modified backprop	2018	99.49%
Zhang [60]	spiking CNN	modified backprop	2019	99.62%
Chankyu Lee [27]	spiking CNN	modified backprop	2020	99.59%
Diehl [35]	ML-SNN	conversion	2015	98.6%
Lee [21]	ML-SNN	modified backprop	2016	98.88%
Wu [58]	ML-SNN	modified backprop	2017	98.89%
Jin [57]	ML-SNN	modified backprop	2018	98.93%

Table 1: different accuracy achievements on MNIST dataset. ML-SNN = Multi-layer spiking neural network

One must remember that due to the many differences in both neural coding and neuron model implementations, it might not be fair to simply list the accuracy results side by side. Much of the progress and research on spiking neural networks do not specifically try to achieve the highest possible accuracy score no matter the cost. One such example is Kulkarni 2018 [26], achieving a respectable 98.17% accuracy on MNIST with a supervised learning method. The interesting part of [26], however, is that the accuracy was achieved while using far less learnable synapses than most other neural networks (Kulkarni used 81.120 learnable synapses compared to Lee 2016 [21] that used 581.520. While, the previous number one achiever on MNIST - DropConnect 2013 [56] used 2.508.470 learnable synapses, to achieve 99.79% accuracy).

Although only comparing the best achieved accuracies on MNIST by spiking neural networks do not give a full picture of the progress made, it should still be somewhat helpful in order to see the direction, as well as rate of improvement for spiking neural network performance.

As to the reason for not including the Esser 2015 [46] and Liu 2017 [48] models in table 1 although they achieved 99.42% and 99.10% respectively as ML-SNNs, is that they were both trained on neuromorphic hardware, compared to the models in table 1, which was all simulated on traditional hardware.

## 5. Benefits of Spiking Neural Networks

In chapter 2, some different spiking neuron models were introduced, and in chapter 3 coding schemes necessary for interpreting spike signals, which is needed for spiking neural networks, was covered. It became clear from those two chapters that spiking neural networks are more complex than conventional artificial neural networks, and as seen in chapter 4, although the gap between traditional neural networks and spiking neural networks is decreasing, spiking neural networks are still behind in terms of accuracy performance. The question is then raised as to why one should consider implementing a spiking neural network, despite the added complexity and inability to achieve performance results on par with traditional neural networks. The current chapter tries to summarize some of the advantages that spiking neural networks pose over conventional artificial neural networks.

### 5.1. Brain Similarity

Much of the motivation behind the development of artificial neural networks is to replicate [23] or, at least to some degree, mimic brain functionality. There are two main reasons for this: 1. To study how the human brain works, and 2. To make artificial neural networks have some or several desirable characteristics similar to the human brain, such as learning ability, adaptivity, generalization ability or low energy consumption [20]. When implementing an artificial neural network for achieving a specified goal, it might not always be ideal to use the most biological plausible model. The choice of neuron model must consider if the added benefits of increased biological plausibility weigh up against the associated computational complexity that follows using a more biologically plausible model. However, a network of biological neurons is able to achieve results and perform tasks well outside the range of what artificial neural networks are capable of. Therefore, one can argue that much is left to be gained by striving for increasing biological plausibility, as it might further increase the ability of artificial neural networks to mimic biological brain functions. Further an argument can be made that each generation of neural networks has seen an increase in the biological plausibility of the networks: The first-generation artificial neural networks implemented the perceptron, introducing synaptic weights that significantly improved the biological plausibility of the network, compared to a network of McCulloch and Pitts neurons. The second-generation neural networks were based on the same neuron model as the first generation, the perceptron, however, the output being interpreted as the firing-rates of the neurons, meant that a notion of time was introduced to the network, albeit only implicitly. The third-generation neural networks consisting of spiking neurons, then implemented time explicitly as a new “dimension” to the system, thereby allowing for asynchronous firing of neurons.

The added biological plausibility in spiking neural networks make them relevant in the study of neuroscience. While spiking neural networks might help study the human brain, it can also go the other way around. Some of the limitations of spiking neural networks have occurred due to a lack of an understanding for processes in the biological brain, such as information encoding and learning. Hence, if advances are made in neuroscience, it is likely that spiking neural networks will also benefit from the discoveries. Therefore, an argument can be made that spiking neural network still has untapped potential.

## 5.2. Computational Power

In order to discuss the computational power of spiking neural networks, it is necessary to define computational power what is meant by computational power. As found by the author, there is mainly two ways one can define computational power, the first is encoding capacity of the network for a given time window, while the second is the number of neurons required to perform a given task. As chapter 2 briefly mentioned the encoding capacity of different coding schemes used by spiking neural networks, this chapter will deal with the second definition of computational power.

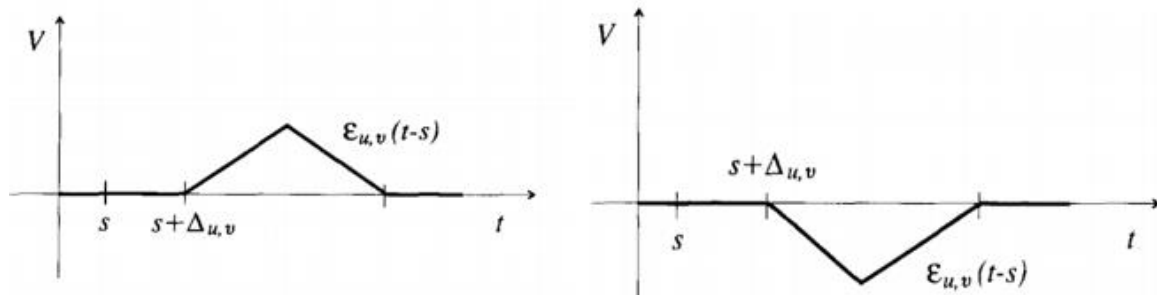


Fig.7: Response functions (EPSP-excitatory post synaptic potential and IPSP - inhibitory post synaptic potential) of a spiking neuron of type B, as defined by Maass. Note: “The particular shape of the ‘triangle’ is not important” [55] (figure taken from [55])

During the 1990s, Maass wrote several papers proving the theoretical computational power of spiking neural networks. In [55] he stated that his work implied that: “networks of spiking neurons (of type B) are in fact *strictly more powerful* than neural nets from the first two generations”. Maass’ statement was based mostly, on a set of findings:

1. Any continuous function  $F: [0,1]^n \rightarrow [0,1]^k$  that can be approximated arbitrarily close with  $s$  sigmoidal units, with activation function  $\pi$  as given by eq.18, can also be approximated arbitrarily close by a network of  $O(s)$  spiking neurons of type B.

$$\pi(y) = \begin{cases} 0, & \text{if } y < 0 \\ y, & \text{if } 0 \leq y \leq 1 \\ 1, & \text{if } y > 1 \end{cases} \quad (\text{eq.18})$$

2. Any given continuous function  $F: [0,1]^n \rightarrow [0,1]^k$  can be approximated by a network of spiking neurons with a single hidden layer.

3. A boolean function  $CD_n: \{0,1\}^{2n} \rightarrow \{0,1\}$  defined by:

$$CD_n(x_1, \dots, x_n, y_1, \dots, y_n) = \begin{cases} 1, & \text{if } x_i = y_i = 1 \text{ for some } i \in \{1, \dots, n\} \\ 0, & \text{otherwise} \end{cases} \quad (\text{eq.19})$$

The boolean function in eq.19 can be computed by a single spiking neuron. In comparison, a neural network of the first generation require at least  $n/\log(n+1)$  threshold gates, while a neural network of the second generation requires  $\Omega(n^{1/4})$  sigmoidal gates in order to compute eq.19.

4. The “element distinctness function” - which takes in an analog input and returns a boolean output -  $ED_n: (\mathbf{R}^+)^n \rightarrow \{0, 1\}$  defined by:

$$ED_n(x_1, \dots, x_n) = \begin{cases} 1, & \text{if } x_i = x_j \text{ for some } i \neq j \\ 0, & \text{if } |x_i - x_j| \geq 1 \text{ for all } i, j \text{ with } i \neq j \\ \text{arbitrary,} & \text{otherwise} \end{cases} \quad (\text{eq.20})$$

The function as given by eq.20 can be computed by a single spiking neuron. In comparison, a neural network of the first generation require  $\Omega(n * \log(n))$  threshold gates in its first hidden layer, while a neural network of the second generation require at least  $\frac{n-4}{2} - 1$  hidden sigmoidal units.

The proof for point 1 - 4 can be found in [55]. From point 1, it follows that spiking neural networks are at least as computationally powerful as neural networks from the two first generations. While point 3. And 4. Shows that spiking neural networks require far less computational units in some specific cases. Leading to Maass’ statement, that spiking neural models are strictly more powerful than earlier generation neural networks.

Important to note is that point 1 through 4 was strictly theoretical results. Appropriate values for the weights in spiking neural networks must be chosen for the points to hold true. As seen in chapter 3, finding appropriate weights by learning in a spiking neural network, proves a harder challenge, compared to earlier generations neural networks.

Further, although spiking neurons are computationally more powerful than their predecessors, their per unit implementation cost is higher than earlier neural models. Therefore, depending on which spiking neuron model is chosen, a network of  $N$  spiking neurons might be more computationally demanding than a network of  $1000 * N$  sigmoidal units.

### 5.3. Energy Efficiency in Event Driven Neural Processing

Energy consumption is a relevant theme in most systems wishing to implement artificial neural networks. For mobile systems, be it robots or phones, the necessity for energy efficiency is clear. However, even for systems that can be plugged in, the benefit of low-energy consumption cannot be underestimated. An increasing amount of energy usage in the world goes toward computing, and it is beneficial to use the minimum amount of energy necessary for any given task, be it for saving money, or the environment.

Neuromorphic hardware is a relatively new form of spike-based hardware. The design principle of such hardware is to ideally have a dedicated processor for each spiking neuron in a network without a central clock [41]. Most neuromorphic hardware in production as of today, such as Intel’s Loihi research chip or SpiNNaker, do virtualize the neurons on their cores – around 1000 for each core, instead of having a one to one mapping between cores and neurons. However, the number of virtualized neurons per core in neuromorphic hardware is far below what one would see in the case of an artificial neural network virtualized on a normal CPU or GPU. Neuromorphic hardware can be seen as a physical implementation of spiking neural networks. Neuromorphic hardware also allows for on-chip learning and enable the neural network implemented on the hardware to be run in real time, increasing the application potential.

The main differences between traditional (von Neumann based) hardware computation and neuromorphic hardware computation can be summarized as in table 2:

<b>von Neumann</b>	<b>Neuro-inspired computing</b>
Very high operation frequency (GHz)	Low operation frequency (KHz)
Centric computation	Distributed computation
Low parallelism	High parallelism
Low power efficiency	High power efficiency

Table 2: Comparison between von Neumann and neuromorphic computing (table taken from [50])



The main reasons for the difference in energy efficiency between the two architectures is due to the difference in operation frequency, as well as the lack of need for a central clock in neuromorphic systems.

One of the main application areas where neuromorphic hardware can see potential use is for low-energy consumption systems that run in real-time, receiving sensory input data from the outside world, such as [14]. The benefit of spiking neural networks in this situation is that spiking neurons are event-driven by nature, meaning that they only compute when receiving an input spike. Hence, for situations where sensory input is sparse, one could expect a significant amount of energy saved, in compared to traditional artificial neural networks that need to update each neuron for every timestep [18]. A further benefit from event-driven computation is that the communication time is reduced greatly. Artificial neurons in traditional neural networks have to send their output value for every timestep. For a spiking neural network, given a certain, small time window, there should only be a small fraction of the spiking neural network sending information. Since only a small portion of the spiking neurons transmit signals at a given time, it results in a better balance between computation of neuron output, and communication time – erasing or limiting communication bottlenecks. The Loihi research chip has been successful in demonstrating a two to three orders of magnitude energy efficiency gain over artificial neural networks implemented on conventional hardware [34].

Neuromorphic chips are, as of today, not available to the consumer market. However, Applied Brain Research (ABR) is in the process of developing what they claim to be an inexpensive neuromorphic board called “Brain Board”, hinted to come out in 2021 in the presentation “Spiking Neural Networks for More Efficient AI algorithms”<sup>2</sup> by Dr. Chris Eliasmith. Dr. Chris Eliasmith is both, the current director of the centre for theoretical neuroscience at the university of Waterloo, and one of the founders of ABR.

### 5.3.1 Pseudo Simultaneous Input-Outputs

Lastly, another benefit from spiking neural networks that run on neuromorphic hardware is that the output from the network start being generated from the first instance of an input signal being received. To further elaborate: When spiking neural networks are run on neuromorphic hardware, allowing for event-driven computation of spike generation, spikes are immediately passed on to the receiving neuron, as the neurons do not need to synchronize their output. The immediate propagation of spikes, down the network, result in the spiking neural networks generating an approximate answer, “pseudo-simultaneously” [62] as the input is received. Hence, the output is generated while the network process the “input event flow” (input that is received over time). The pseudo-simultaneously nature of the input and output flow means that the event-driven spiking neural network can, at each moment, have an estimated “best guess” of what the output is going to be. The importance

---

<sup>2</sup> <https://www.youtube.com/watch?v=PeW-TN3P1hk&t=336s>

of such approximate outputs can be best seen in applications needing fast processing, such as autonomous bipedal robots. The approximate output would allow such a robot to make estimated guesses, based on already received and processed information. Therefore, if the robot started to lose balance, the system would not have to wait for sensory information on say, the hardness of the ground, or head orientation compared to the rest of the body, before starting to perform actions in order to regain balance. Then, when the sensory information on the ground hardness, or relative head orientation becomes available, the system can update its preferred actions to keep balance if necessary.

## 6 Challenges of Spiking Neural Networks

Some of the challenges spiking neural networks face might have already become apparent in earlier chapters. In general, one can argue that spiking neural networks have been limited partly from its non-standardization. A plethora of different spiking neuron models, and neural encoding schemes, make spiking neural networks able to adapt more freely, to give the desired biological plausibility and computational complexity. However, the fact that no general-purpose supervised learning algorithm exist for spiking neural networks can be partly contributed to a lack of spiking neuron model and neural encoding scheme uniformity. The other important factor for the lack of a general-purpose learning algorithm, is the discontinuous nature of the output spike-train. However, if one assume that a leaky-integrate and fire spiking neuron model is used, as well as limiting the encoding scheme options used in order to give feedback to the network, supervised learning algorithms that use both the temporal and rate information present in spike train outputs exist, such as [27].

### 6.1 The State of Spiking Neural Networks

Spiking neural networks are still seen as a niche of artificial intelligence research [54]. Many are highly sceptic of the potential use of spiking neural networks, such as Yann LeCun, chief AI scientist at Facebook, stating “I’m very sceptical about this (neuromorphic hardware and spiking neural networks)”<sup>3</sup> at ISSCC 2019 (International Solid-Sate Circuits Conference). However, others, such as Michael Frank, chief architect at Arteris IP have said “SNNs have their domain where they will outrun a standard network. Even a digital emulation of an SNN is better than a classical CNN” [33]. Some of the disagreement might stem from a disagreement of what constitutes a spiking neural network. Disagreement in the definition of spiking neural networks, can lead to people having quite different views about the state of spiking neural networks as of today. Some claim spiking neural networks are far from being commercialized and maybe never will, while others, especially those working on spiking neural networks themselves, have a more generous outlook, such as Intel’s Loihi team and ABR researchers. As spiking neural networks are still regarded as a relatively niche area of artificial intelligence research, statements from prominent individuals in the field of AI, such as Yann LeCun, can make people less inclined to actually investigate on the state of spiking neural networks any further.

---

<sup>3</sup> <https://www.youtube.com/watch?v=YzD7Z2yRL7Y&feature=youtu.be&t=2120>

## 6.2 Comparability

Another challenge for spiking neural networks is the benchmarks, such as MNIST, used to compare spiking neural networks with other artificial neural networks. Benchmarks such as MNIST does not enable spiking neural networks to showcase many of their associated benefits, such as low-energy consumption or reduced latency. Such benchmarks also require spiking neural networks to perform some sort of input-signal mapping. Second-generation neural networks can easily convert images to numbers for neural input, based on the pixel greyscale. However, the way to convert an image into an input for a spiking neural network is not that easy, as one need to implement the notion of time in the input signal. The conversion from image to spike-train input is typically lossy and inefficient [41]. The inability to show the potential benefits of spiking neural networks through conventional benchmarks, might also be some of the reason for the difference in view of the state of spiking neural networks.

Further, as stated before, these benchmarks might not be a fair way of comparing spiking neural networks against each other, as different networks use different neural models and neural encoding schemes, which cannot be compared in directly in a fair way. Hence, it is difficult to accurately see the progress of development of spiking neural networks.

## 6.3 Non-Optimal Hardware

the existing von Neuman based hardware available today is not optimized for spiking neural networks. The lack of available neuromorphic hardware has meant that much of the benefits one can expect from spiking neural networks is yet to been seen, mostly associated with temporal event-driven processing. There exist only a limited number of loihi-based test systems<sup>4</sup>, which means getting hold of one for research might be a challenge. For what the author could find, the other neuromorphic systems existing today also only exist in limited quantities. Although, neuromorphic hardware availability will change in the near future for the better, the lack of optimal hardware to-date might be a reason for a slowed down progress in the development and research of spiking neural networks.

---

<sup>4</sup> <https://www.intel.com/content/www/us/en/research/neuromorphic-computing.html>

## 7 Conclusion

More than two decades have passed since Maass wrote his influential paper “Networks of Spiking Neurons: The third Generation of Neural Network Models” introducing the concept of spiking neural networks as the third generation artificial neural network, while laying the foundation of further research into the topic, by theoretically proving their computational capacity. Nevertheless, spiking neural networks have yet to replace the second-generation neural networks, sigmoidal neural networks, as the main workhorse of the artificial neural network community. From what the author could gather, this is due to several facts:

Firstly, sigmoidal neural networks are already really efficient in a range of tasks such as still-image recognition. Further, as new learning methods have to be developed to be used in spiking neural networks, it is hard to overcome sigmoidal neural networks that has benefitted from many extra years of research and development.

Secondly, although sigmoidal neural networks were mainly just an upgrade compared to the earlier threshold neural networks, spiking neural networks cannot be seen as a strict upgrade to sigmoidal neural networks in the same way. Spiking neural networks and sigmoidal neural networks excel at different areas. Spiking neural networks will most likely find its usage in real-time processing, such as self-driven cars and robots needing to respond differently depending on sensory input, learning while actively performing tasks. Other areas of use are low-power units, where the tiny gains in extra accuracy expected from sigmoidal neural networks is of little importance, compared to the sometimes-massive energy saving benefit that neuromorphic hardware can bring.

Thirdly, as spiking neural networks are more biological plausible than sigmoidal neural networks, problems such as how to interpret spike-trains as an output suddenly arise. Machines today use almost no effort to compare value outputs and give the answer as the highest number. However, as there is still no conclusive theory about neural encoding, several different ideas are implemented, which are more costly to interpret than simple number values. The neural coding problem adds a new area of discussion, where spiking neuron models can implement different strategies, making spiking neural networks less uniform than their earlier generation artificial neural network counterparts.

Then as to answer the research question of whether spiking neural networks will replace sigmoidal neural networks; It is the authors belief that spiking neural networks will most likely not replace sigmoidal neural networks the same way sigmoidal neural networks replaced their predecessor, at least in the foreseeable future. Newfound research in the field of neuroscience on how the brain learns, or encodes information might change this, however this is not a topic the author is able give any constructive remarks on. Although the author does not believe that spiking neural networks will replace sigmoidal neural networks, the author still believes that spiking neural networks certainly has its place in the years to come. Spiking neural networks will most likely see usage in real-time event-driven applications, as well as other temporal-associated tasks, such as speech processing. Further,

spiking neural networks can serve as a way to study the processes in biological neural networks.

As the belief of the author is that spiking neural networks will not replace sigmoidal neural networks, but instead be developed alongside it, the second research question become obsolete. Nevertheless, the author will take the opportunity to talk about what has held spiking neural networks back in general. There seem to be two main reasons that has held back spiking neural networks thus far.

The higher biological plausibility of spiking neural networks has meant that implementing learning methods have become harder as one suddenly has to deal with the temporal axis and are back to discontinuous outputs. There is reason to believe that this will remain a challenge, until a breakthrough in the understanding of how the biological neural network learns and encodes information occur.

Further another reason for spiking neural networks still being considered a niche might be due to the lack of available neuromorphic hardware, which can showcase the benefits of spiking neural networks. Applications imagined to benefit from spiking neural networks, such as self-driving cars, and many forms of more advanced robots needing fast processing for such as balancing, are still in the research phase and under constant development. While sigmoidal neural networks see their deployment in several areas targeting consumers such as social media filtering, user-targeted advertisement, and language translation, spiking neural networks are still mainly implemented related to research only. The lack of examples of successful commercial implementations of spiking neural networks so far, might have limited the amount of investment spiking neural networks have seen to date. Although, it seems that interest for spiking neural networks are growing, based on the amount of research papers being generated last year and during this spring (2020).

## 7.1 Further Work

Spiking neural networks touch upon several different fields of expertise, mainly focusing on machine learning and neuroscience. Future work expanding the survey include reading up on different forms of learning methods not mentioned in chapter 4, mostly Reinforcement learning methods, as reinforcement learning has become an increasingly important area of learning in the machine learning community in recent years.

Further, another way of training spiking neural networks not touched upon in this survey is the idea of reservoir computing. From what the author could gather the main idea behind reservoir computing is based around the recurrent neural network topology. The reservoir computing topology consist of a set of input cells, a recurrent network, consisting of a set of neurons with randomly generated connections between them in a sparse and fixed manner, as well as a set of output neurons. The input and output neurons are connected to the recurrent neural network, which is called the “reservoir” (hence the name reservoir computing). The output neurons read the state of the reservoir at a given time and map it to

an output signal. What differentiates reservoir computing from recurrent neural networks is that the reservoir is usually treated as an “unapproachable” unit left untrained, so the only training required are of the output weights. The idea of only training the output weights makes supervised training of artificial networks implementing reservoir computing, easier compared to implementing supervised training on traditional recurrent neural networks. Reservoir computing is not limited to spiking neural networks, however, spiking neural networks allow for a form of STDP learning (unsupervised) to be implemented on the reservoir itself, without needing to understand the dynamics of the reservoir.

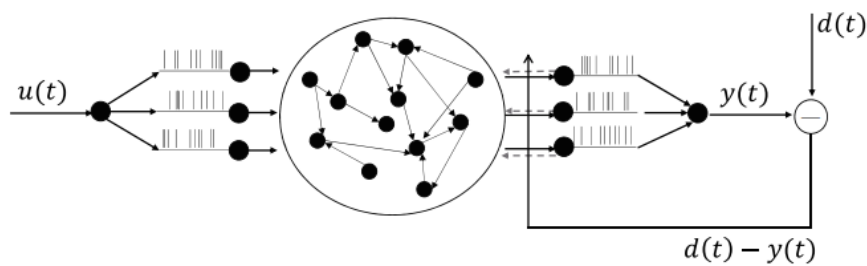


Fig.8 representing reservoir computing on a spiking neural network.  $d(t) - y(t)$  is the error signal used for supervised training of the output weights, while the circle containing randomly interconnected neurons are the reservoir. (figure taken from [59])

Reservoir computing can be divided into Echo state network (ESN), which can be used by non-spiking neural networks as well as spiking neural networks, and liquid state machines (LSM), reserved for spiking neural networks. More research is needed by the author to gain a thorough understanding of the subject of reservoir computing in spiking neural networks. [65] Use LSM to perform automatic speech recognition.

Lastly, another specific neuron model worth exploring further, not introduced in this survey, is the spike response model (SRM). The main difference between the SRM model and the other models mentioned in this survey is that the firing threshold of the neuron in question is not gained as an input, but instead time-dependent based on the firing history of the neuron. Interestingly enough, it is apparently possible to map an integrate and fire neuron model to an SRM model [63]. More investigation of the underlining mathematics behind the SRM model is needed by the author for it to be included in this survey. Hence, it is instead recommended as an opportunity of further reading for those interested.

## 8 References

- [1] A. Hodgkin and A. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, pp. 500–544, 1952.
- [2] B. Rueckauer, Y. Hu, I.-A. Lungu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, p. 682, 2017
- [3] Bi, G. Q., & Poo, M. M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 18(24), 10464–10472. <https://doi.org/10.1523/JNEUROSCI.18-24-10464.1998>
- [4] Bouvier, Maxence & Valentian, Alexandre & Mesquida, Thomas & Rummens, Francois & Reyboz, Marina & Vianello, E. & Beigne, Edith. (2019). Spiking Neural Networks Hardware Implementations and Challenges: A Survey. *ACM Journal on Emerging Technologies in Computing Systems*. 15. 1-35. 10.1145/3304103.
- [5] Brette, Romain. "Philosophy of the Spike: Rate-Based vs. Spike-Based Theories of the Brain." *Frontiers*, Frontiers, 10 Nov. 2015, [www.frontiersin.org/articles/10.3389/fnsys.2015.00151/full](http://www.frontiersin.org/articles/10.3389/fnsys.2015.00151/full).
- [6] Brunel, Nicolas & van Rossum, Mark. (2007). Quantitative investigations of electrical nerve excitation treated as polarization: Louis Lapicque 1907 · Translated by: *Biological Cybernetics*. 97. 341-349. 10.1007/s00422-007-0189-6.
- [7] Burkitt, Anthony. (2006). A Review of the Integrate-and-fire Neuron Model: I. Homogeneous Synaptic Input. *Biological cybernetics*. 95. 1-19. 10.1007/s00422-006-0068-6.
- [8] Byerly, Adam & Kalganova, Tatiana & Dear, Ian. (2020). A Branching and Merging Convolutional Network with Homogeneous Filter Capsules.



- [9] Chapter 4 Spiking Neural Model. Maynooth University Computer Science Department, [cortex.cs.may.ie/theses/chapters/chapter4.pdf](http://cortex.cs.may.ie/theses/chapters/chapter4.pdf).
- [10] D.O. Hebb. *The Organization of Behaviour*. Wiley, New York, 1949.
- [11] Dan, Yang, and Mu-Ming Poo. "Spike Timing-Dependent Plasticity: From Synapse to Perception." *Physiological Reviews*, 1 July 2006, [journals.physiology.org/doi/full/10.1152/physrev.00030.2005](http://journals.physiology.org/doi/full/10.1152/physrev.00030.2005).
- [12] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, Nov 2003.
- [13] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, Sept 2004.
- [14] F. Galluppi *et al.*, "Event-based neural computing on an autonomous mobile platform," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, 2014, pp. 2862-2867, doi: 10.1109/ICRA.2014.6907270.
- [15] Gerstner, W., Kistler, W. Mathematical formulations of Hebbian learning. *Biol Cybern* 87, 404–415 (2002). <https://doi.org/10.1007/s00422-002-0353-y>
- [16] Gerstner, W., Kistler, W., Naud, R., & Paninski, L. (2014). *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge: Cambridge University Press. doi:10.1017/CBO9781107447615
- [17] Gordon, Dan. "Brain is 10 Times More Active than Previously Measured, UCLA Researchers Find." UCLA, UCLA, 9 Mar. 2017, <https://newsroom.ucla.edu/releases/ucla-research-upend-long-held-belief-about-how-neurons-communicate>
- [18] Grüning, André, and Sander M. Bohte. *Spiking Neural Networks: Principles and Challenges*. 2014, [homepages.cwi.nl/~sbohte/publication/es2014-13Gruning.pdf](http://homepages.cwi.nl/~sbohte/publication/es2014-13Gruning.pdf).
- [19] Hagan, Martin T. *Neural Network Design*. 2nd ed., Martin Hagan, 2014.

- [20] Jain, Anil K, et al. "Artificial Neural Networks: A Tutorial." LSU.edu, 1996, [csc.lsu.edu/~jianhua/nn.pdf](http://csc.lsu.edu/~jianhua/nn.pdf).
- [21] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in Neuroscience*, vol. 10, p. 508, 2016.
- [22] Kang, Byeong Ho, and Quan Bai. *AI 2016: Advances in Artificial Intelligence 29th Australasian Joint Conference, Hobart, TAS, Australia, December 5-8, 2016: Proceedings*. Springer, 2016.
- [23] Khan, Gul Muhammad. *Evolution of Artificial Neural Development: in Search of Learning Genes*. Springer, 2018.
- [24] Kheradpisheh, Saeed Reza & Ganjtabesh, Mohammad & Thorpe, Simon & Masquelier, Timothée. (2017). STDP-based spiking deep convolutional neural networks for object recognition. *Neural Networks*. 99. 10.1016/j.neunet.2017.12.005.
- [25] Knudsen EI (1994) Supervised Learning in the Brain. *J Neurosci* 14: 3985–3997
- [26] Kulkarni, Shruti & Rajendran, Bipin. (2018). Spiking neural networks for handwritten digit recognition—Supervised learning and network optimization. *Neural Networks*. 103. 10.1016/j.neunet.2018.03.019.
- [27] Lee C, Sarwar SS, Panda P, Srinivasan G and Roy K (2020) Enabling Spike-Based Backpropagation for Training Deep Neural Network Architectures. *Front. Neurosci.* 14:119. doi: 10.3389/fnins.2020.00119
- [28] LeVine, Steve. "Artificial Intelligence Pioneer Says We Need to Start Over." *Axios*, 15 Dec. 2017, [www.axios.com/ai-pioneer-advocates-starting-over-2485537027.html](http://www.axios.com/ai-pioneer-advocates-starting-over-2485537027.html).
- [29] Loiselle, Stephane & Rouat, Jean & Pressnitzer, Daniel & Thorpe, Simon. (2005). Exploration of Rank Order Coding with Spiking Neural Networks for Speech Recognition. 4. 2076 - 2080 vol. 4. 10.1109/IJCNN.2005.1556220.

- [30] Luo, X., Qu, H., Zhang, Y., & Chen, Y. (2019). First Error-Based Supervised Learning Algorithm for Spiking Neural Networks. *Frontiers in neuroscience*, 13, 559. <https://doi.org/10.3389/fnins.2019.00559>
- [31] Mehlig, B. "Artificial Neural Networks." *Arxiv*, 1 Feb. 2019, [arxiv.org/pdf/1901.05639.pdf](https://arxiv.org/pdf/1901.05639.pdf).
- [32] Meng, Mingyuan & Yang, Xingyu & Xiao, Shanlin & Yu, Zhiyi. (2020). Spiking Inception Module for Multi-layer Unsupervised Spiking Neural Networks.
- [33] Moyer, Bryon. "Spiking Neural Networks: Research Projects or Commercial Products?" *Semiconductor Engineering*, 18 May 2020, [semiengineering.com/spiking-neural-networks-research-projects-or-commercial-products/](https://semiengineering.com/spiking-neural-networks-research-projects-or-commercial-products/).
- [34] Nandakumar, S.R., Boybat, I., Le Gallo, M. *et al.* Experimental Demonstration of Supervised Learning in Spiking Neural Networks with Phase-Change Memory Synapses. *Sci Rep* **10**, 8080 (2020). <https://doi.org/10.1038/s41598-020-64878-5>
- [35] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Neural Networks (IJCNN)*, 2015 International Joint Conference on. IEEE, 2015, pp. 1–8.
- [36] Pan, Zihan, et al. "Neural Population Coding for Effective Temporal Classification." *Arxiv*, Department of Electrical and Computer Engineering National University of Singapore, 26 Sept. 2019, [arxiv.org/pdf/1909.08018.pdf](https://arxiv.org/pdf/1909.08018.pdf).
- [37] Panda, Priyadarshini, et al. "Asp: Learning to forget with adaptive synaptic plasticity in spiking neural networks.", *arXiv preprint arXiv:1703.07655v2*
- [38] Park, Seongsik, et al. "Fast and Efficient Information Transmission with Burst Spikes in Deep Spiking Neural Networks." *Arxiv*, 10 Feb. 2019, [arxiv.org/pdf/1809.03142.pdf](https://arxiv.org/pdf/1809.03142.pdf).

- [39] Park, Seongsik, et al. "T2FSNN: Deep Spiking Neural Networks with Time-to-First-Spike Coding." Arxiv, Department of Electrical and Computer Engineering, ASRI, INMC, and Institute of Engineering Research Seoul National University, 26 Mar. 2020, [arxiv.org/pdf/2003.11741.pdf](https://arxiv.org/pdf/2003.11741.pdf).
- [40] Paugam-Moisy, H el ene & Bohte, Sander. (2012). Computing with Spiking Neuron Networks. 10.1007/978-3-540-92910-9\_10.
- [41] Pfeiffer, M., & Pfeil, T. (2018). Deep Learning With Spiking Neurons: Opportunities and Challenges. *Frontiers in neuroscience*, 12, 774. <https://doi.org/10.3389/fnins.2018.00774>
- [42] Ponulak, Filip & Kasiński, Andrzej. (2011). Introduction to spiking neural networks: Information processing, learning and applications. *Acta neurobiologiae experimentalis*. 71. 409-33.
- [43] Rosenblatt, F.F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6, 386-408.
- [44] Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. *Nature* 323, 533–536 (1986). <https://doi.org/10.1038/323533a0>
- [45] S. Bohte, J. Kok, et al. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48:17–38, 2002.
- [46] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Advances in Neural Information Processing Systems*, 2015, pp. 1117–1125.
- [47] Song S, Miller KD, Abbott LF. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity // *Nat Neurosci.*, 2000, 3(9), :pp. 919-926.

- [48] T. Liu, Z. Liu, F. Lin, Y. Jin, G. Quan, and W. Wen, "Mt-spike: a multilayer time-based spiking neuromorphic architecture with temporal error backpropagation," in Proceedings of the 36th International Conference on Computer-Aided Design. IEEE Press, 2017, pp. 450–457.
- [49] Tavanaei, Amirhossein, et al. "Deep Learning in Spiking Neural Networks." Arxiv, 20 Jan. 2019, [arxiv.org/pdf/1804.08150.pdf](https://arxiv.org/pdf/1804.08150.pdf).
- [50] The, Vu & Ben Abdallah, Abderazek. (2019). Algorithms and Architectures for Spiking Neuromorphic Systems. 10.13140/RG.2.2.23714.79049.
- [51] Thorpe, Simon & Gautrais, Jacques. (1998). Rank Order Coding. Computational Neuroscience: Trends in Research. 113-118. 10.1007/978-1-4615-4831-7\_19.
- [52] Tuckwell, H. C. (1988). Introduction to theoretical neurobiology. Vols. 1 and 2. Cambridge: Cambridge University Press.
- [53] VanRullen, Rufin & Guyonneau, Rudy & Thorpe, Simon. (2005). Spike times make sense. Trends in neurosciences. 28. 1-4. 10.1016/j.tins.2004.10.010.
- [54] von Kugelgen, Julius. (2017). On Artificial Spiking Neural Networks: Principles, Limitations and Potential.
- [55] W. Maass. Networks of spiking neurons: The third generation of neural network models. Neural Networks, 10:1659–1671, 1997
- [56] Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., & Fergus, R. (2013). Regularization of Neural Networks using DropConnect. In Proceedings of machine learning research: vol. 28. Proceedings of the 30th international conference on machine learning (pp. 1058–1066). Atlanta, Georgia, USA: PMLR, URL <http://proceedings.mlr.press/v28/wan13.html>.
- [57] Y. Jin, P. Li, and W. Zhang. Hybrid macro/micro level backpropagation for training deep spiking neural networks. arXiv preprint arXiv:1805.07866, 2018.

- [58] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. arXiv preprint arXiv:1706.02609, 2017.
- [59] Zhang, Anguo & Zhu, Wei & Liu, Ming. (2017). Self-Organizing Reservoir Computing based on Spiking-Timing Dependent Plasticity and Intrinsic Plasticity Mechanisms. 10.1109/CAC.2017.8243892.
- [60] Zhang, Wenrui & Li, Peng. (2019). Spike-Train Level Backpropagation for Training Deep Recurrent Spiking Neural Networks. arXiv:1908.06378
- [61] Zhang, L., Zhou, S., Zhi, T., Du, Z., & Chen, Y. (2019). TDSNN: From Deep Neural Networks to Deep Spike Neural Networks with Temporal-Coding. *AAAI*.
- [62] Camuñas-Mesa L. A., Serrano-Gotarredona T., Linares-Barranco B. (2014). Event-driven sensing and processing for high-speed robotic vision, in Proceedings of the 2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) (Lausanne: BioCAS; ), 516–519
- [63] Gerstner, Wulfram, et al. *Neuronal Dynamics: from Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2015.
- [64] McCulloch, W.S., Pitts, W. “A logical calculus of the ideas immanent in nervous activity.” *Bulletin of Mathematical Biophysics* 5, 115–133 (1943).  
<https://www.cs.cmu.edu/~epxing/Class/10715/reading/McCulloch.and.Pitts.pdf>
- [65] Verstraeten, D., Schrauwen, B., & Stroobandt, D. (2005). Isolated word recognition using a Liquid State Machine. *ESANN*.

