TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Mariam Bokeria 156298

CHARACTER RECOGNITION WITH SPIKING NEURAL NETWORK

Master's thesis

Supervisor: Eduard Petlenkov
Associate Professor

Tallinn 2017

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mariam Bokeria 156298

SÜMBOLITE TUVASTAMINE IMPULSSNÄRVIVÕRKUDEGA

magistritöö

Juhendaja: Eduard Petlenkov
Associate Professor

Tallinn 2017

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Mariam Bokeria

04.05.2017

Abstract

This work aims to show the image, in this case English uppercase letters, to be recognized by spiking neural network. Different types of neural network models are introduced, which are modeled for neural network and pattern recognition. For this case of work Izhikevich model is used and implemented in MATLAB. Finally, as a conclusion and result the solution to the image processing problem is shown.

This thesis is written in English and is 45 pages long, including 8 chapters, 19 figures and 0 tables.

Annotatsioon

Sümbolite tuvastamine impulssnärvivõrkudega

Töö eesmärgiks on spiking-tüüpi närvivõrkude tööprintsiipide ja kujundite tuvastamise jaoks rakendatavuse uurimine. Töös käsitletakse erinevaid impulssi närvivõrkude struktuure. Töös on demonstreeritud Ižikevitši mudel realisatsioon MATLABis ning see on rakendatud kujundite tuvastamise ülesande lahendamiseks. Meetodi rakendamine on demonstreeritud tähtede tuvastamise näitel.

Töö on kirjutatud inglise keeles ning sisaldab teksti 45 leheküljel, 8 peatükki ning 19 joonist.

List of abbreviations and terms

SNN	Spiking Neural Network
IF	Integrate-and-Fire
LIF	Leaky Integrate-and-Fire
STDP	Spike-Timing-Dependent Plasticity
LTP	Long Term Potentiation
LTD	Long Term Depression
SAPR	Synaptic Activity Plasticity Rule
RGB	Red, Green, Blue color model

Table of contents

List of figures

1 Introduction

This paperwork is about image or character recognition using an artificial spiking neural network. The goal of this work is to present historical background of the artificial neural networks in computer science, name some existing famous models of neural networks and descuss examples. The problem to be highlighted is image or character wich must be recognized using spiking neural network (SNN). This paperwork mostly contains the survey about SNN, but the problem to be solved is English upper case letters of alphabet which must be recognized by the network.

The first section is going to be the introduction, the second section will be the brief history of neural networks, the third section will be about existing neural network models, it will contain the subsections about some famous models (McCulloch-Pitts, Hodgkin-Huxley-type model, Integrate-and-fire (IF), Leaky Integrate-and-fire (LIF) and Izhikevich models). In the fourth section, there will be descussed some neural coding techniques. The next fifth section will be about network learning rules. After that in the sixth section there will be shown the solution to the main problem defined above. Finally, as a conclusion and summery there will be the main goal presented and shown the answers to the problem stated in here.

Before starting to introduce the SNN let's compare the brain and the computer, artificial and biological neuron [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15].

The main differences between the brain and the computer are:

- The basic building blocks of the brain are biological neurons and they are slower than silicon logic gates. The neurons operate in milliseconds while the silicon gates operate in the nanosecond range which is about six orders of magnitude faster.

- The brain makes up for the slow rate of operation with two factors: one is that it has huge amount of neurons and interconnections between them. The human brain

contains approximately 10^{14} to 10^{15} interconnections. The ohter thing is that the function of biological neuron is much more complex than the logic gate.

- The brain is very energy efficient. It consumes about 10-16 joules per operation per second while a digital computer consumes 10-6 joules per operation per second.

- Brain is a highly complex, non-linear, parallel information processing system. It performes tasks like: pattern recognition, preception, motor control much more faster than the digitaal computers.

- If we consider the efficiency of a visual system like a complex task of perceptual recognition for example a recognition of a familiar face can be accomplished in 100-200 ms while tasks of much less coplexity takes hours or even days on conventional computers.

Here are introduced the differences between atrificial and biological neuron:

Speed: neural networks are faster in processing information than biological neurons. The cycle time corresponding to execution of one step of a programm in the central processing unit is in the range of few nano seconds while for biological neuron it takes milliseconds.

Processing: many programs have large number of instructions and they operate in a sequential mode, one instruction after another on a conventional computer while biological neural networks can perform massively parallel operations.

Size and complexity: The size and complexity of connections gives the brain the power of performing complex pattern recognition tasks, which can not be realized on a computer.

Storage: In a computer the information is stored in a memory which is addressed by its location. Any new information in the same location can destroy the old information. Here it is strictly replaceable. Biological neurons store information in the strength of interconnections. Information in the brain is adaptable because the new information is added by adjusting the interconnection strengths without destroying the old information.

Fault tolerance: Artificial nets are not fault tolerant, since the information corrupted in the memory can not be retrieved while in biological neurons if few connections are not working the information is still preserved due to the distributed nature of the encoded information.

Control mechanism: In computer there is a control unit which monitores all the activities of computing while in the brain there is no central control for processing information. The neuron acts based on the information locally available and transmits its output to the neurons connected to it. So there is no specific control mechanism.

After defining the main differences between artificial and biological neurons, let's introduce SNN.

Biological neurons use short or sudden increases in voltage to send the information. These signals are called spikes or pulses.

Humans have sensors all over the body. We constantly receive sensory inputs from the environment, we process the information, we can recognize danger, food, subjects, etc. and act according to that information. Not only humans are like this, but animals and everything that interacts with its environment needs to do so.

Millions of neurons are interconnected with each other. They cooperate to efficiently process incoming signals and decide on actions. In fact, scientists still don't understand completely how the single neuron is functioning, but it is understood how they are working. Neurons send out short pulses, spikes as signals. Basically this was used and mathematicaly modeled in computer use, as it's called artificial neural networks.

Artificial neural networks are already becoming a fairly old technique within computer science. There are three generations of artificial neurons. The first idea and model is over fifty years old. The first generation of artificial neural networks was a very simple model conceptually: a neuron sends a binary signal if the sum of its weighted incoming signals rises above a threshold value. This kind of neurons have been successfully applied in powerful artificial neural networks like multi-layer perceptrons. For example, any function with Boolean output can be computed by a multilayer perceptron with a single hidden layer. These networks are called universal for digital computations.

Neurons of the second generation do not use a step- or threshold function to compute their output signals, but a continuous activation function, making them suitable for analog in- and output. Commonly used examples of activation functions are the sigmoid and hyperbolic tangent. Typical examples of neural networks consisting of neurons of these types are feed-forward and recurrent neural networks.

Neuron models of the first two generations do not employ individual pulses, but their output signals typically lie between 0 and 1.

In the third generation of neural networks neurons use pulse coding instead of using the rate coding. For example humans analyse and classify visual input (i. e. facial recognition) in under 100 ms. This leaves about 10 milliseconds of processing time per neuron. Such a time-window is very little to allow an averaging mechanism like rate coding. This does not mean that rate coding is not used, but when speed is an issue pulse coding schemes are favoured.

Computers communicate with bits; neurons use spikes. Incoming signals change the voltage of the neuron and when this reaches above a threshold-value the neuron sends out an action potential itself. Such an action potential is a short (1ms) and sudden increase in voltage that is created in the cell body or soma. Due to their form and nature we refer to them as spikes or pulses [16] [17].

SNN models are the third generation of neural networks and are considered to be one of the most biologically accurate models [18].

Figure 1 shows the example of neural networks with spiks visualy, which was prvided in [19].

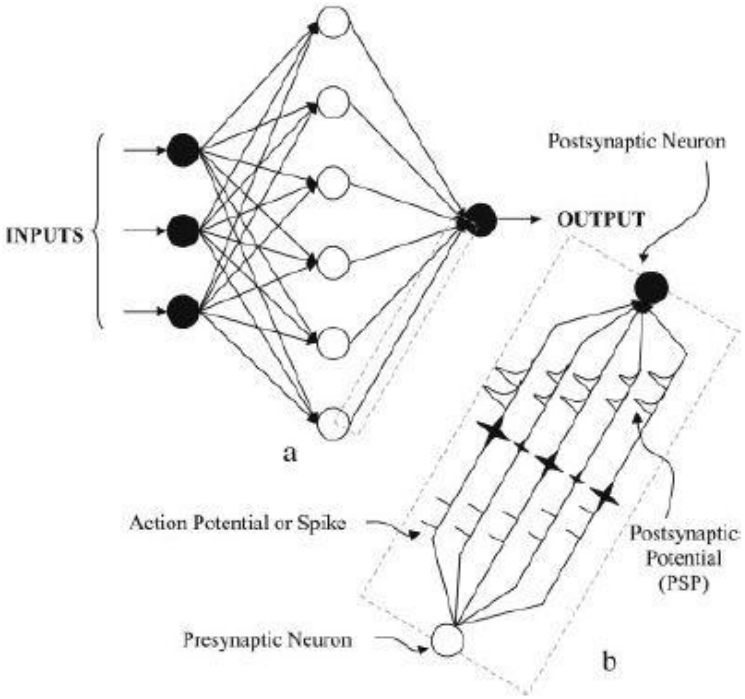

Figure 1. Neural network with spike

2 A Brief History Of Neural Networks

The study of human brain dates back thousands of years. The recent advances in science (such as electronics, cognitive and computer science) have allowed us to partially emulate the human brain and its innate cognitive ability.

Here, in this section, some of the most significant neural network designs and models are introduced.

In 1943, Warren McCulloch and Walter Pitts designed and built a primitive artificial neural network using simple electric circuits that formed the basis for modern era of neural network research.

With the emergence of computers in the 1950s, the neural models were ported from hardware to the digital realm. Alan Lloyd Hodgkin and Andrew Huxley described a scientific model of a spiking neuron in 1952. They explained the ionic mechanisms underlying the initiation and propagation of action potentials in the squid giant axon.

Hodgkin-Huxley model is widely regarded as one of the great achievements of 20th-century biophysics that describes how action potentials in neurons are initiated and propagated.

Later, in 1954 Marvin Minsky carried out research on neural networks.

In 1958, based on the idea of McCulloch-Pitt's theory and research done, a neurobiologist named Frank Rosenblatt worked on the idea of perceptron. He built the first artificial neural network realized in hardware.

In 1960, Bernard Wildrow and Marcian Hoff developed the Adaptive Linear Neuron or later known as Adaptive Linear Element and Multiple Adaptive Linear Neuron models which were the first neural networks applied to real problems. Adaptive Linear Element is a convergent type single layer neural network based on the McCulloch-Pitts neuron consisting of a weight, a bias and a summation function. It's used for prediction involving binary patterns as its input and one output. Similarly, for problems requiring multiple outputs, multiple sets of Adaptive Linear Elements are used in parallel and this model is known as Multiple Adaptive Linear Neuron.

First introduced by Bryson and Ho (1975), the Backpropogation neural networks gained recognition through the work of David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams in 1986. They extended the Widrow and Hoff's delta rule to networks with multiple hidden layers by means of generalized delta rule. This model was based on the perceptron and came to be known as the Back Propagation Network, which requires differentiable activation (transfer) function. It is one of the most widely used artificial neural network model.

In 1990 Jeff Elman presented a simple recurrent neural network, which is a feed-forward network modified by one or more feedback connections.

More recently spiking neural networks (Gerstner & Kistler, 2002; Izhikevich, 2003) have been developed. It belongs to the third generation of neural networks. Their mechanism is more realistic in terms of their spiking processes resembling the biological neurons [20].

3 Neuron Models

Here, in this section some famous neuron models will be descussed. Spiking neural network consists of many interconnected neurons. So as previously descussed in the above section the history of artificial neural network already starts from the beginning of the 20th century. I've chosen some famous neuron models to name and to talk about.

3.1 McCulloch-Pitts Model

The McCulloch-Pitts model of neuron is one of the earliest and simplest model which was firstly introduced n 1943. It is also known as linear threshold gate. It is a neuron of a set of inputs $I_1, I_2, I_3, \dots, I_m$ and one output y. The inputs could be either a zero or one and the output also zero or one. Also each input could be either excitatory or inhibitory. So the linear threshold gate simply classifies the set of inputs into two different classes and the output y is binary. Such a function can be described mathematically using following equations:

$$Sum = \sum_{i=1}^{N} I_i W_i \quad (1)$$

$$y = f(Sum) \quad (2)$$

$W_1, W_2, W_3, \dots, W_m$ are weight values normalized in the range of either (0, 1) or (-1, 1) and associated with each input line, Sum is the weighted sum. The function f is a linear step function at threshold value [21].

The whole point of this model is to sum the inputs. If the the input is one and is excitatory it added one, if the input is again one and is inhibitory it substructed one from the sum. This is done for all the inputs and the final sum is calculated.

If this final sum is less than some threshold value, then the output is zero. Otherwise the output is one.

3.2 Hodgkin-Huxley-type model

As known from [22] Reference Hodgkin-Huxley neuron model which was firstly introduced in 1952 represents a relatively high degree of biological accuracy describing

neuron function. This model has been widely and successfully used in neuroscience research [22].

The Hodgkin–Huxley model is considered to be one of the most biologically accurate spiking neuron models. It consists of four differential equations and a large number of parameters. The differential equations describe the neuron membrane potential, activation of Na and K currents, and inactivation of Na currents. The model can exhibit almost all types of neuronal behavior if its parameters are properly tuned. This model is very important to the study of neuronal behavior and dynamics as its parameters are biophysically meaningful and measurable. A time step of 0.01 ms was utilized to update the four differential equations as this is the most commonly used value [18].

$$\frac{dv}{dt} = \left(\frac{1}{C}\right)\{I - g_k n^4(V - E_k) - g_{Na} m^3 h(V - E_{Na}) - g_L(V - E_L)\} \quad (3)$$

$$\frac{dn}{dt} = (n_\infty(V) - n)/\tau_n(V) \quad (4)$$

$$\frac{dm}{dt} = (m_\infty(V) - m)/\tau_m(V) \quad (5)$$

$$\frac{dh}{dt} = (h_\infty(V) - h)/\tau_h(V) \quad (6)$$

3.3 Integrate-and-fire (IF) model

The most widely used and best-known model of thresholdfire neurons, and spiking neurons in general, is the integrate-and-fire neuron.

Once the voltage over the capacitor goes above threshold value ϑ the neuron sends out a pulse itself. Mathematically we write:

$$\tau_m \frac{\partial u}{\partial m} = -u(t) + RI(t) \quad (7)$$

To describe the effects on membrane potential u over time, with τ_m being the membrane time constant in which voltage „leaks“ away. As with the spike-response model the neuron fires once u crosses threshold ϑ and a short pulse δ is generated. To force a refractory period after firing we set u to $K < 0$ for a period of δ^{abs}.

$$I_i(t) = \sum_{j\epsilon\Gamma_i} c_{ij} \sum_{t_j^{(f)}\epsilon F_j} \delta(t - t_j^{(f)}) \quad (8)$$

The input current I for neuron i will often be 0, as incoming pulses have a finite short length. Once a spike arrives, it is multiplied by synaptic efficacy factor c_{ij} forming the postsynaptic potential that charges the capacitor. This model is computationally simple and can easily be implemented in hardware. It is closely linked to the more general spike-response model and can be used like it by rewriting it into the correct kernels η and ε [17].

3.4 The Leaky Integrate-and-Fire Neuron Model

According to [23] research article I found out that the leaky integrate-and-fire neuron is one of the simplest spiking neuron models which is still very popular. In the simplest form the neuron is modeled as a "leaky integrator" of its input $I(t)$:

$$\tau_m \frac{dv}{dt} = -v(t) + RI(t) \quad (9)$$

where $v(t)$ represents the membrane potential at time t, τ_m is the membrane time constant and R is the membrane resistance. This equation describes a simple resistor-capacitor (RC) circuit where the leakage term is due to the resistor and the integration of $I(t)$ is due to the capacitor that is in parallel to the resistor. The spiking events are not explicitly modeled here but when the membrane potential $v(t)$ reaches a certain spiking threshold v_{th} value, it is reset to a lower value v_r (reset potential) and the leaky integration process described by first Equation starts a new with the initial value v_r. Also it is possible to add an absolute refractory period Δ_{abs} immediately after $v(t)$ hits v_{th}. During the absolute refractory period, membrane potential $v(t)$ might be clamped to reset potential v_r and the leaky integration process is re-initiated following a delay of absolute refrectoriness Δ_{abs} after the spike.

Stimulation by a constant input current: If considering the case with constant input current $I(t) = I$ and $v_r = 0$ then the solution to first given equation will be this:

$$v(t) = RI[1 - \exp\left(-\frac{t}{\tau_m}\right)] \quad (10)$$

The asymptotic value of the membrane potential is RI. If this value is less than the spiking threshold, v_{th}, no spike can be generated. If, however, $RI > v_{th}$, then the neuron generates spikes. If $v(0) = v_r = 0$, the time of the first spike $t^{(1)}$ can be found like this:

$$v_{th} = RI[1 - \exp\left(-\frac{t^{(1)}}{\tau_m}\right)] \quad (11)$$

$$t^{(1)} = \tau_m ln \frac{RI}{RI - v_{th}} \quad (12)$$

This equation also shows the time between each successive spike the neuron fires. But this is only because of the case when $v(0) = v_r$. If adding the absolute refrectory period Δ_{abs} also then the equation will be:

$$T = \Delta_{abs} + \tau_m ln \frac{RI}{RI - v_{th}} \quad (13)$$

f is the mean firing rate of the neuron and it is given by $1/T$:

$$f = \left[\Delta_{abs} + \tau_m ln \frac{RI}{RI - v_{th}}\right]^{-1} \quad (14)$$

Stimulation by a time-varying input current: For a general time-varying input current $I(t)$ the solution to the main first equation will be:

$$v(t) = v_r \exp\left(-\frac{t - t_0}{\tau_m}\right) + \frac{R}{\tau_m}\int_0^{t-t_0} \exp\left(-\frac{s}{\tau_m}\right) I(t - s)ds \quad (15)$$

Where the initial condition $v(t_0) = v_r$. This equation also describes the dynamics of the membrane potential between successive spiking events. When the membrane potential reaches the threshold, it is immediately reset to v_r and starts to evolve according to the equation again until the next spiking event.

Stimulation by synaptic currents: Previous examples considered the stimulation of the Leaky Integrate-and-Fire neuron by the direct injection of constant or time-varying currents. Now, consider a more realistic situation where the neuron is stimulated by pre-synaptic spikes arriving at its synapses. Each pre-synaptic spike makes a stereotyped contribution, described by a function $\alpha(t)$, to the post-synaptic current and contributions of different pre-synaptic spikes are linearly summed to obtain the total post-synaptic current.

The total post-synaptic current to the i-th neuron is this:

$$I_i(t) = \sum_j \omega_{ij} \sum_f \alpha(t - t_j^{(f)}) \quad (16)$$

Where $t_j^{(f)}$ represents the time of the f-th spike of the j-th pre-synaptic neuron; ω_{ij} is the strength of synaptic efficacy between neuron i and neuron j. Common choices for α include the instantaneous Dirac δ-pulse: $\alpha(t) = q\delta(t)$, where q is the total charge injected into the synapse;

The alpha synapse:

$$\alpha(t) = \alpha \frac{t}{\tau} \exp\left(1 - \frac{t}{\tau}\right) \quad (17)$$

The bi-exponential synapse:

$$\alpha(t) = \beta \frac{\tau_2}{\tau_2 - \tau_1} \left[\exp\left(-\frac{t}{\tau_1}\right) - \exp\left(-\frac{t}{\tau_2}\right)\right] \quad (18)$$

Where α and β are normalizing constants and τ, τ_1 and τ_2 are the time constants of the synapses [23].

3.5 Izhikevich model

Izhikevich, has developed a class of models of spiking neurons that balances the computational efficiency of Integrate-and-Fire models with the biological plausibility and versatility of Hodgkin-Huxley type models.

By generating sequences of action potentials, nerons process data. Neurons encode computations into sequences of spikes which are biophysically determined by the cell's action-potential-generating mechanism. Izhikevich proposed a simplified model, which contains two coupled differential equations, but is able to reproduce complex neural behaviour. The model is based on following equations:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I \quad (19)$$

$$\frac{du}{dt} = a(bv - u) \quad (20)$$

with the auxiliary after-spike resting equations:

$$v \geq V_{th}\ then \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \quad (21)$$

Here, v represents the membrane potential of the neuron and u represents a membrane recovery variable, which accounts for the activation of K^+ ionic currents and inactivation of Na^+ ionic currents, and it provides negative feedback to v. After the spike reaches the threshold value V_{th} the membrane voltage and the recovery variable are reset according to the equations above. If v skips over V_{th}, then it first is reset to V_{th} and then to c so that all spikes have equal magnitudes. The part $0.04v^2 + 5v + 140$ is chosen so that v is in the mV scale and time is in ms. If we rewrite the Izhikevich model, here we have:

$$\dot{u} = a(u - u_r)(u - u_t) + I \quad (22)$$

If $u \geq u_{peak}$ then $u \leftarrow u_{reset}$. It can be simplified by applying a first-order Euler approximation to the quadratic model:

$$u_{k+1} = u_k + I_k + a(u_k - u_r)(u_k - u_t) \quad (23)$$

If $u_{k+1} \geq u_{peak}$ then $u_{k+1} \leftarrow u_{reset}$; where k is the step number. The last term can be approximated using two taps:

$$u_{k+1} = u_k + I_k + \left\lfloor \frac{u_k}{P_2} \right\rfloor + \left\lfloor \frac{u_k}{P_2} \right\rfloor \quad (24)$$

Where $P_i = (-1)^{s_i} * 2^{p_i}$ with p_i and s_i being the parameters of the i^{th} tap. On the other hand, in equilibrium (without input), the Izhikevich model is:

$$u = 0.004v^2 + 5v + 140 \quad (25)$$

$$u = bv \quad (26)$$

These equations, from a geometrical viewpoint, represent a parabolic curve and a line. The crossing points of these curves give the equilibrium points of the system (neuron). Different spiking patterns are produced by changing these crossing points and the threshold action potential. The crossing points can be calculated as $E_1 = (e_{v1}, e_{u1})$ and $E_2 = (e_{v2}, e_{u2})$, where [14]:

$$e_{v1,2} = 12.5\left((b - 5) \pm \sqrt{b^2 - 10b + 2.6}\right) \quad (27)$$

$$e_{u1,2} = be_{v1,2} \quad (28)$$

4 Neural Coding Techniques

The system composes of 3 functional parts: the encoding part, the learning part and the readout part.

Each part performs different functional role in the system: the encoding one generates the set of specific activity patterns that represent the various attributes of external stimuli; the learning one tunes the neurons' weights making sure particular neurons can respond to certain patterns correctly; the readout part extracts information about the stimulus from a given neural response. Through this architecture, the problem of getting data into and out of the spiking neural network is solved, and the task of pattern recognition could be fulfilled.

The aim of the encoding part is to generate spiking patterns that represent the input stimuli. The encoding neuron has M input points which are selected from the components of the stimulus. It performs a specific function to convert the input points into latencies within the encoding window. For example, if the stimulus is composed of binary values (0 or 1), the function of the encoding neuron is to convert the binary string into a decimal value.

The learning part of network is composed of one layer of tempotrons. The encoding neurons are fully connected to the learning neurons. The number of encoding neurons is determined by the number of patterns. The tempotron can perform the classification task as long as the load is less than a critical value. Therefore, as long as the number of patterns does not exceed the critical load value, the network can perform the task well. If there are too many patterns, the number of encoding neurons should be increased correspondingly.

The tempotrons comprise the learning neurons. The learning neuron fires or not when it is presented to a stimulus and then the synaptic efficacies are updated according to the learning rule.

The aim of the readout part is to extract information about the stimulus from the response of learning neurons. Each learning neuron responds to a stimulus by firing (1) or not firing

(0). So, the total N learning neurons as the output can represent a maximum number of 2^N classes of patterns. The number of learning neurons is determined by the number of classes in the recognition task. For example, four readout is sufficient for a group of patterns containing 16 classes [24].

In subsections below there are some coding techniques discussed and in the next Section – learning rules.

4.1 Input Encoding

Spiking neural networks differ significantly from early neural networks. The presence and precise timing of spikes encapsulates meaning. Therefore different techniques are required to submit a stimulus to the network. Here I will name and descuss techniques of transforming data into a suitable form for network submission [22].

4.2 Rate Coding

The notion of rate coding assumes that a significant portion of information is encoded in the firing rate or frequency of neurons. Probabilistic firing rates can be used to encode information. This technique of encoding has been criticized for several reasons. In particular, behavioral experiments show that human response times to visual stimuli are very short, which would not leave enough time for an average firing rate to be determined by the system [22].

4.3 Sine Wave Encoding

In the supervised classification problem there exists input features which must be transformed to an acceptable format for the spiking neural network. One method of transformation is sine wave encoding. The raw feature values are normalized and then the amplitude of the sine wave is adjusted based on the normalized feature value. This signal is presented to the network for some portion of the total simulation time. Since the amplitude of the signal is encoding the information this technique is very similar to the continuous inputs of traditional neural networks [22].

4.4 Spike Density Encoding

A spike density code is a form of population coding that measures how many neurons are firing. In other words, a pool of neurons could be set up so that neurons fire stochastically relative to the size of the input value. Therefore, the density of the spikes generated by the pool as an entire unit encodes the input information. One issue with this method is the apparent inefficiency of using such a large number of neurons to encode a relatively few number of inputs [22].

4.5 Temoporal Encoding

Temporal coding is a way of encoding input information as time differentials. This technique may also be called latency coding or time-to-first-spike coding. This technique takes advantage of the spiking neural networks ability to encode information temporally [22].

Pattern recognition is a general task that assigns an output value to a given input pattern. The first step for pattern recognition is to understand how the information is stored in a pattern. How the information is represented in the brain still remains unclear. However, there is a strong evidence showing that using pulses to encode, as a basic means of information transfer, is optimal in terms of information transmission. Two basic and widely studied coding schemes using these pulses are rate coding and temporal coding. In rate coding the number of spikes within a time window is considered while the precise timings of each spike are considered in temporal coding. Temporal patterns in the spike train can carry more information than rate-based coding. A simple example of temporal encoding is spike latency code.

Temporal learning rule aims on dealing with information encoded by precise timing spikes. One of the most commonly studied rules is the spike-timing-dependent plasticity (STDP) which has emerged in recent years as experimentally most studied form of synaptic plasticity. According to STDP learning rule, the plasticity depends on the intervals between pre- and postsynaptic spikes. The basic mechanisms of plasticity found in STDP are the long term potentiation (LTP) and the long term depression (LTD). However, STDP characterizes synaptic changes solely in terms of the temporal contiguity of the presynaptic spike and the postsynaptic potential or spike [24].

4.6 Rank Order Encoding

Coding by rank order is a technique where the order of the spikes is used to encode information. Such a coding scheme would require the mapping of input values to a rank order over n neurons. The spike emissions are among one of the n! possible orderings of n neurons. Therefore, $\log_2(n!)$ bits may be used to represent such an ordering. Such a capacity is optimistic as using this method within computer simulations necessitates the ability to differentiate between two spike timings [22].

The neurons within rank order coding scheme are considered as "integrate-and-fire" devices. A neuron integrates its inputs over time until it reaches a threshold, and fires a single action potential. The neuron is then reset and after a certain refractory period starts integrating information again. Based on this property of neurons we can assume that the **firing rate** of neuron is a monotonous **function** of the strength of its input. The neural system needs a relatively long time to stabilize the firing rate of neurons.

Since the firing rate of neuron is a monotonous function of the strength of its input, it is natural to assume that neuron with highest input would firstly arrive the threshold value and fire a spike. In other words, the latency of firing of a neuron also reflects the strength of its input. We can conclude that the input of the neuron with shorter latency of firing is higher than the neuron with longer latency of firing [1].

5 Learning Rules

Biologically founded neural networks like spiking neural networks are capable of self-learning from their input. The network's behavior is shaped by inputs that it has received over time. Here we can have a look at two different synaptic plasticity rules that may be used for networks of spiking neurons [22].

5.1 Synaptic weight modification

A common method used to modify the connection weights is based on the observations of Konorski and Hebb. Such synaptic weight modifications are termed plasticity. The

Konorski/Hebbian learning rule changes the weight of a synaptic connection based upon the pre- and post-synaptic neuron activity. When the firing of a pre-synaptic neuron regularly participates in the firing of a post-synaptic neuron, the strength of the action of the pre-synaptic neuron onto the post-synaptic neuron increases. If the pre-synaptic neuron regularly fires after the post-synaptic neuron, the strength of the action from the pre-synaptic neuron onto the post-synaptic neuron decreases [22].

5.2 Spike Timing-Dependent Plasticity

Spike timing-dependent plasticity (STDP) is a temporally asymmetric form of Konorski/Hebbian learning that modifies synaptic connections between pre- and post-synaptic neurons that are temporally correlated. In addition, a winner takes all approach is often used where only the weight of the first post-synaptic neuron to fire is updated. Such plasticity is believed to be an underlying learning and information storage mechanism and possibly contributes to the development of neuronal circuits during brain development [22].

5.3 Synaptic Activity Plasticity Rule

The Synaptic Activity Plasticity Rule (SAPR) is a temporally symmetric form of Konorski/Hebbian learning. The synaptic connection strength in SAPR is modified using an update function that takes advantage of the membrane potential of the post-synaptic neuron. In contrast to the STDP function, SAPR is continuous when the time difference between pre- and post-synaptic firing times is zero, where STDP is undefined. Values for STDP approach $\pm\infty$ as the time difference nears zero, whereas SAPR is bounded to a finite range [22].

6 Character Recognition – Review Analyses

What is pattern recognition? Given input image is first sense and then segmentation of image is done. For each segment of image features are extracted and then classification is done which include adjustment for missing feature. Then post processing is done which

include adjustment for missing context and cost. Finally decision is made by comparison of evaluating parameters such as average firing rate, accuracy, efficiency, simulation time for image processing in pattern recognition [19].

So, to define pattern recognition we can say that it is a part of machine learning that focuses on the recognition of patterns in data. In other words it is a process of classifying input data into objects or classes based on key features.

For this work English uppercase letters are used to be recognized with SNN. According to studies SNN is one of the layers of feed forward neural network. It is also known as the third generation of artificial neural network. It is a two layered structure. There are both, excitatory and inhibitory connections. In general, There are three broad classes of neurons based on their effect on other neurons' membrane potential: excitatory, inhibitory and modulatory neurons [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40].

In pattern recognition there are two classification methods: supervised and unsupervised classification. In previous sections I generaly descussed about neural coding techniques and Learning rules. Later, I will continue to talk about the case which I've chosen for this work (English uppercase letters).

Figure 2 shows the basic structure of SNN for character recognition which was provided in [2].

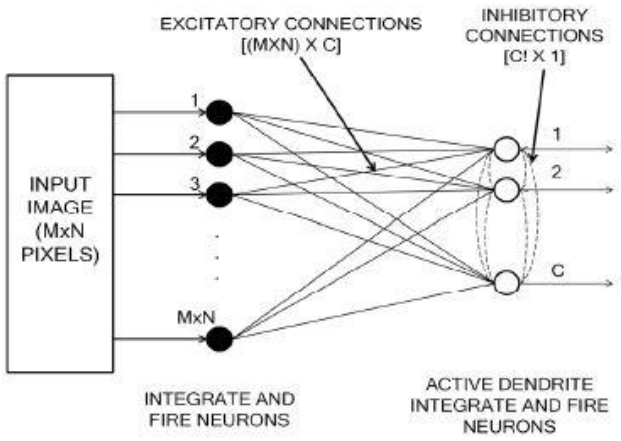

Figure 2. Spiking Neural Network for character recognition

In our case the example which is provided for this work, as I already mentioned, are English uppercase letters. Figure 3 shows how the input layer of one letter looks like.

```
letterA =  [0 0 1 0 0 ...
            0 1 0 1 0 ...
            0 1 0 1 0 ...
            1 0 0 0 1 ...
            1 1 1 1 1 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ]';
```

Figure 3. Input Letter "A"

For all the other letters we have similar input matrices. Here is another example of letter "B" (see Figure 4).

```
letterB =  [1 1 1 1 0 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 1 1 1 0 ...
            1 0 0 0 1 ...
            1 0 0 0 1 ...
            1 1 1 1 0 ]';
```

Figure 4. Input letter "B"

For solving the letter recognition problem I've chosen Izhikevich model of spiking neurons. In one of the sections above (3.5) I talked about Izhikevich model. This model is more simple and computationally efficient than Integrate-and-Fire (IF) or other neuron models. This mathematical model was proposed by Eugene Izhikevich in 2003 and it is the most recent of models used to study individual neurons that display spiking/bursting behavior.

Izhikevich model is pretty much interesting because it is compact model and suitable change of some parameters can simulate a large array of neurons behavior.

As shown in section 3.5 the Izhikevich model consists of two differential equations. There are two main variables: the membrane potential v (in millivolts mV) and the generic recovery variable u. There are also five free parameters in the equation: I – the external input to the neuron; a – the rate of recovery of u; b – the sensitivity of recovery to subthreshhold fluctuations of membrane potential; c, d are the after spike resets of v and u.

Later in Appendix 1 is shown MATLAB code provided for this paperwork which solves the letter recognition problem. The code is written according to Izhikevich model which uses Euler method for computations.

As I already showed in section 3.5 the Izhikevich model consists of a system of two differential equations:

$$v' = 0.04v^2 + 5v + 140 - u + I \quad (29)$$

$$u' = a(bv - u) \quad (30)$$

With the auxiliary after-spike resetting:

$$v \geq V_{th}\ then \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \quad (31)$$

Here v and u are dimensionless variables and a, b, c and d are dimensionless parameters.

The model does not have a fixed threshold but after the spike reaches +30 mV the membrane voltage v and the recovery variable u are reset. The synaptic current is delivered via the variable I.

The membrane potential v has mV scale and the time t has ms scale. The value of the parameter $a = 0.02$. It describes the time scale of the recovery variable u. The smaller the value, the slower the recovery.

The parameter b describes the sensitivity of the recovery variable u to the subthreshold fluctuations of the membrane potential v. A typical value is chosen $b = 0.2$. The greater the values v and u, the stronger is the result in spiking dynamics.

The parameter c describes the after-spike reset value of the membrane potential v and the value is chosen $c = -65$ mV. The parameter d describes the after-spike reset of the recovery variable u and the value is chosen $d = 2$.

There exist different types of spiking reactions. For different images spiking responses or reactions can be different. As already mentioned the given input images for the network are English letters. I've chosen first two letters "A" and "B" images for my first experiment.

First I've set each image size as small as possible and I took 35x26 pixels size because the main variable "Rec_record" in the code (see Appendix 1) to be smaller or in other

words to say, the amount of input neurons to be less. The simulation time step “t” equals 200 and the amount of input neurons for each image is 910 (in case 35x26 pixels size).

So by changing the parameters of “Rec_record”, in other words, by setting the amount of neurons in the last part (lines) of the code we can observe the spike reactions to each input image.

There is also one important fact about this experiment - the RGB colors. For this paperwork the chosen input images are black and white. So while plotting the spike responses for image we choose the parameters of input neurons from 0 to 910 for white color and for black. Mostly the spike response is type of fast spiking.

Figure 5 shows how the input image of letter “A” looks like.

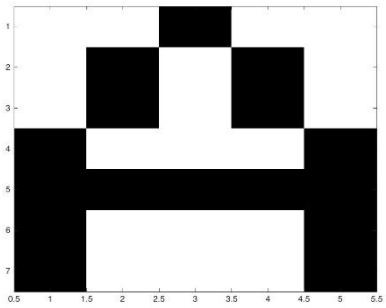

Figure 5. English uppercase letter “A”

Figure 6 shows how the input image of letter “B” looks like.

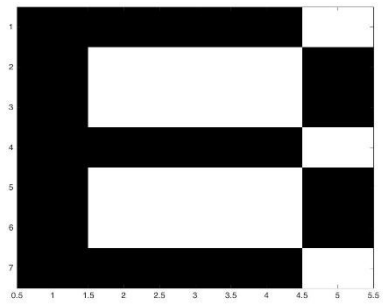

Figure 6. English uppercase letter “B”

So, as I already said, I’ve chosen the image to be 35x26 pixels size. Firstly, for letter “A” I’ve chosen parameters 10 and 400. As I already explained this are amount of input neurons for white and black color. Figure 7 shows the result of recognized letter.

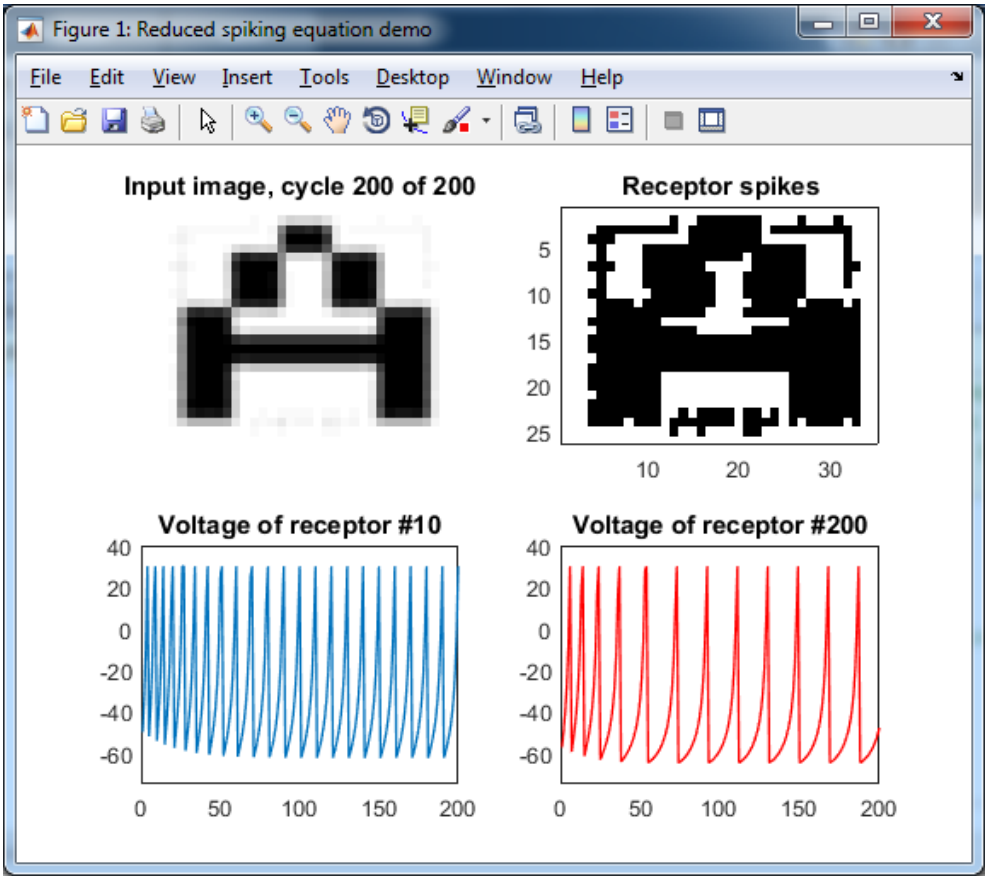

Figure 7. As a result the recognized letter “A” and spike reactions

As you can see, there is a letter “A” recognized by the network and different spike reactions to the image. The spike reaction to the function “Rec_record (10, 1:t)” is shown in Figure 8.

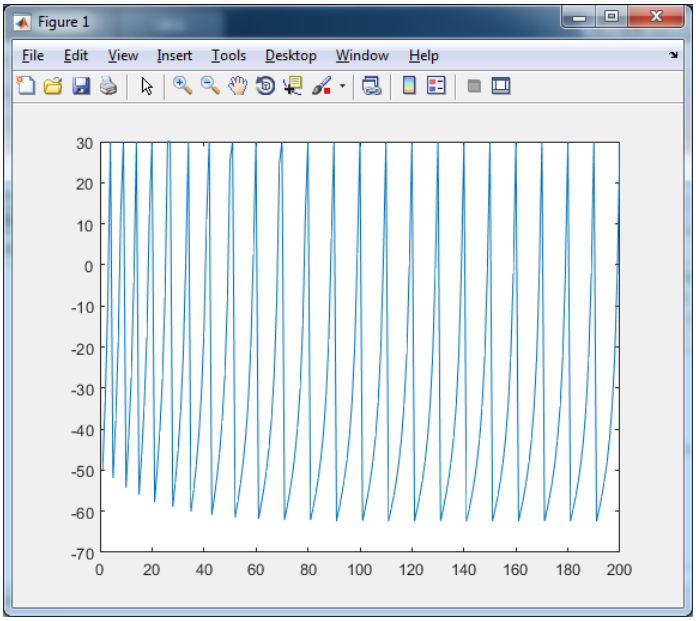

Figure 8. Spike reaction to the input amount of neurons (10)

Figure 9 shows the spike reaction to the function “Rec_record (400, 1:t)”.

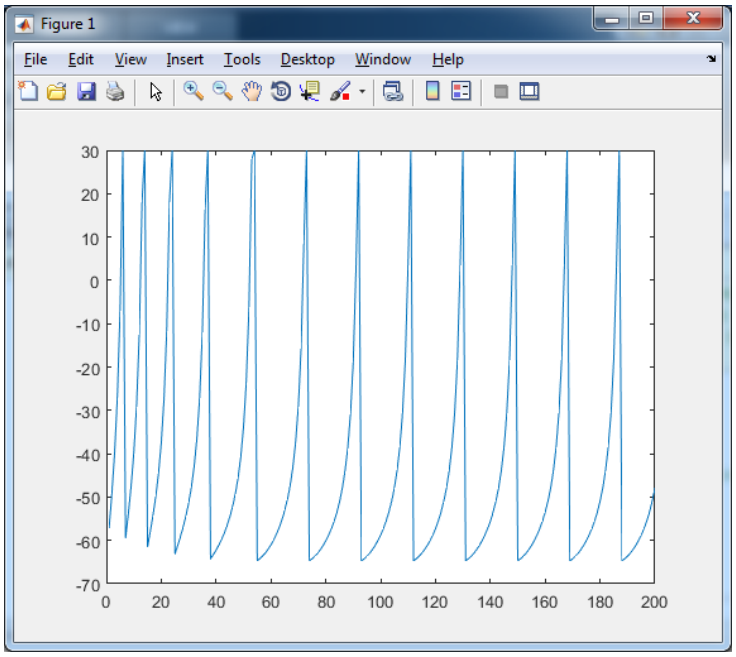

Figure 9. Spike reaction to the input amount of neurons (400)

As you can see there is a difference between neuron reactions to the image. The result shown in Figure 8 is type of fast spiking and the result shown in Figure 9 is type of low-threshold spiking. This result, spike reaction, depends on which neurons are activated. In this case it is considered that each pixel of the image is one neuron. So the reaction depends on which pixels we take for the experiment, like now it was taken 10 and 400.

Let’s see another example of experiment for the letter “B”. We can observe the recognized letter “B” on the Figure 10.

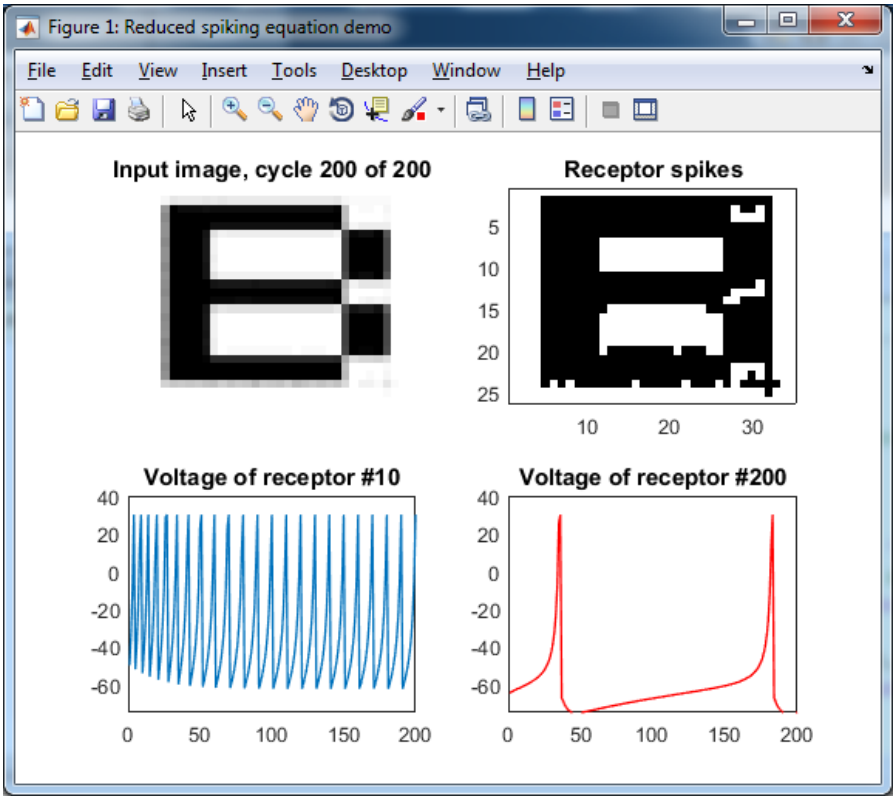

Figure 10. As a result the recognized letter “B” and spike reactions

So, as we can see on Figure 10 the letter was recognized but the spike reactions to the mage are very different. For this experiment next parameters, pixels were chosen 10 and 795. Figure 11 shows the spike reaction to the function “Rec_record (10, 1:t)”.

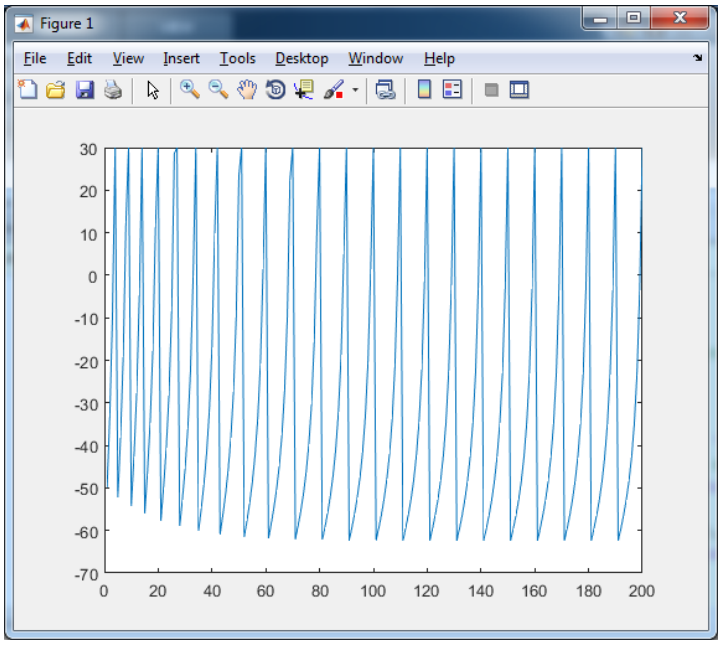

Figure 11. spike reaction to the input amount of neurons or input pixels (10) for letter “B”

Figure 12 shows the spike reactions to the input pixels or same as input neurons of which amount is 795.

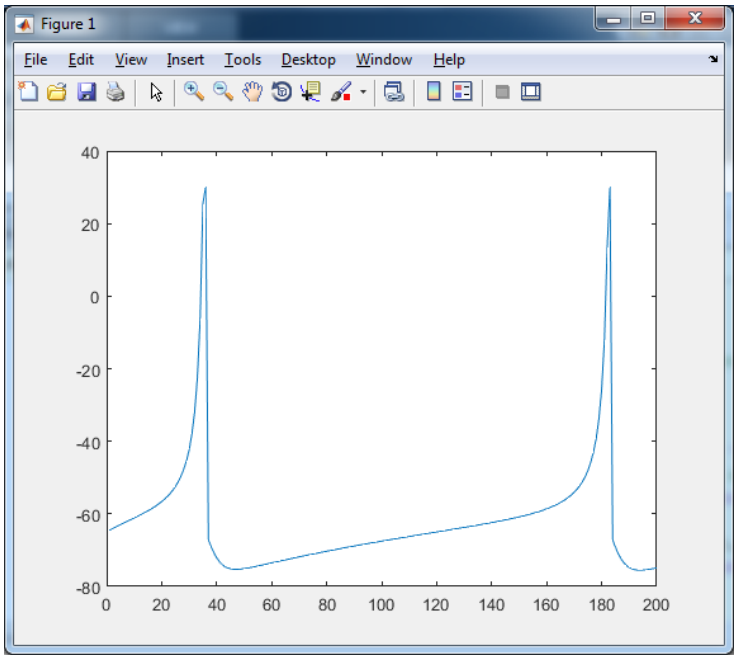

Figure 12. spike reaction to the input amount of neurons or input pixels (795) for letter “B”

As you can observe, there is also a big difference between spike reactions depending on which pixels are taken for the experiment. The reaction shown on Figure 11 is again fast spiking type and the reaction shown on Figure 12 looks more like regular spiking type.

So, we can observe different behaviors of neurons and their reactions for different kinds of images choosing different input neurons, in our case same as pixels.

Besides these experiments, the main problem is to somehow analyse the neurons and get the final result as recognized image. On the output it should be obvious weather the letter or image was recognized or not.

The script provided in Appendix 1 was taken from Izhikevich model and it only shows the graphycal part, spike reactions, but still it is unclear if the image was recognized. As mentioned above the image taken was the size of 35x26 which means that there are 910 pixels or same as 910 neurons which are pretty hard to analyse. Also the simulation time step which is 200 is hard to analyse.

So for the next experiment I took the simulation length of time step 50 but as for image it would be easier to analyse if the image was 5x7 pixels of size because in this case we would have only 35 neurons.

I took again letters “A” and “B” and applied the same algorithm to see the spike reactions. Figure 13 shows the result for letter “A”.

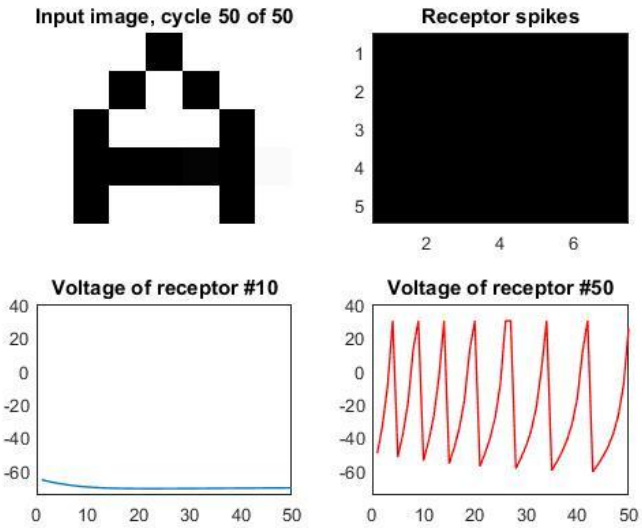

Figure 13. Spike reactions to the image “letter A” the size of 5x7

Figure 14 shows the result, spike reactions for letter “B”.

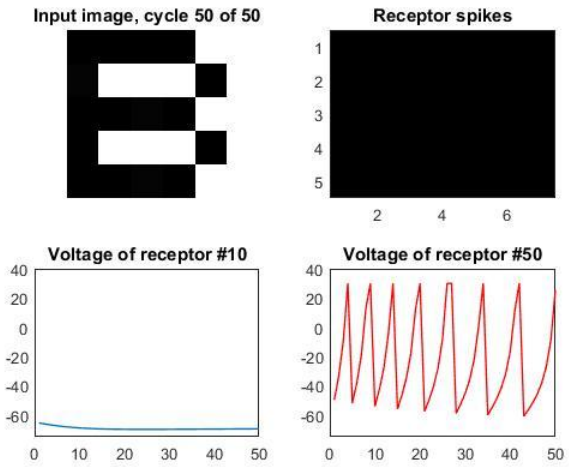

Figure 14. Spike reactions to the image “letter B” the size of 5x7

As I mentioned once, the image is black and white color. So, whatever neurons we take from 0 to 35 (as we have 5x7 image size and 35 neurons) the reaction will be according to color. For instance, the blue line on the figures 13 and 14 shows the reaction of the neuron for the white color.

After visual result of spike reactions let’s see the result as matrice for the receptor’s membrane potential.

25.40078	25.40078	25.40078	-70.5832	25.40078	21.38163	25.40078
19.42585	23.37316	-70.5832	25.40078	-70.5832	25.40078	25.40078
25.40078	-70.5832	25.40078	25.40078	25.40078	-70.2538	21.38163
23.37316	-70.5832	-70.2538	-70.5832	-69.1858	-70.5832	17.50544
25.40078	-70.5832	25.40078	25.40078	21.38163	-69.5569	25.40078

Figure 15. The result matrice. Recognized letter “A”

As you can see on the figure 15 there is a matrice which shows the recognized letter “A”. As the image was the size of 5x7 it is easy to analyse that the pixels or the same as nurons for the black color are highlighted in red here and the number or the meaning of those pixels are approximately -70. Same result of the receptor’s membrane potential is shown for the letter “B” on figure 16.

25.40078	-70.5832	-69.5569	-70.5832	-70.5832	25.40078	25.40078
25.40078	-69.1858	21.38163	23.37316	25.40078	-70.5832	25.40078
23.37316	-70.5832	-70.2538	-69.1858	-70.5832	25.40078	25.40078
25.40078	-70.5832	25.40078	19.42585	23.37316	-70.5832	21.38163
21.38163	-69.5569	-70.5832	-69.1858	-70.2538	25.40078	25.40078

Figure 16. The result matrice. Recognized letter “B”

The highlighted pixels with numbers of approximately -70 are again for black color. So. We can easily see the recognized letters on the matrices of receptor’s membrane potential. We could see the same matrice of the receptor’s recovery variable, but there instead of -70 the meaning of highlighted or black pixels would be approximately -13.

There is another experiment that I tried. I took the image and added the noise and then I applied the same algorithm for recognition. Figure 17 shows the image with noise.

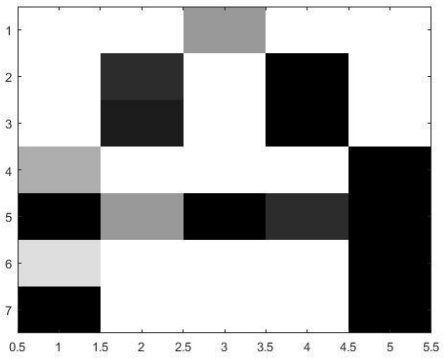

Figure 17. Letter A with noise

For this experiment I took again the image of letter “A” with noise with the size of 5x7.

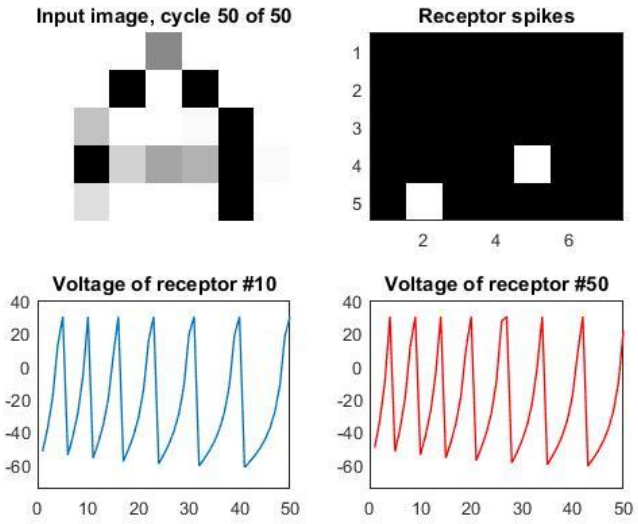

Figure 18. Spike reactions to the image of letter "A" with noise

Now for the image with noise the colors are no more only black and white, as you can see there is a little bit of different kinds of grey. So, the figure 18 shows the spike reactions to the image. The recognized result shown again with matrice as receptor's memrane potential is presented on figure 19.

19.42585	25.40078	25.40078	-31.9021	19.42585	25.40078	21.38163
25.40078	25.40078	-70.5832	25.40078	-70.2538	25.40078	19.42585
25.40078	-55.9605	25.40078	25.40078	17.50544	-70.5832	25.40078
25.40078	-70.5832	-2.02718	-51.429	44.5832	-69.9123	15.61999
23.37316	88.49144	25.40078	21.38163	21.38163	-70.2538	25.40078

Figure 19. Recognized image of letter "A" with noise

As you can see the areas where pixels were black is still recognized with approximate number -70 and the less highlighted areas with blur grey color have less meaning.

7 Conclusion

In this paperwork were presented the brief history of artificial neural network and how it was developed during the years, survey about spiking neural network (SNN) and the way it works. Different kinds of neural models were presented starting from the very first, the simplest McCullogh-Pitts perceptron model to the latest Izhikevich spiking neuron

model, also including Integrate-and-fire (IF) and the other neuron models. There were theoretically introduced several neural coding techniques and learning rules.

Later, it was the simplest Izhikevich model taken, using Euler method caclulations which was implemented in MATLAB for image recognition experiment. The model consists of the simple system of two differential equations. For the images English alphabet was used, uppercase letters. The experiment was presented using MATLAB environment. The input neurons behavior and their reactions during simulated time steps were observed. Each pixel of given image was considered to be one neuron. Depending on which pixel or neuron to choose as parameter of a function we could observe the reaction of the chosen neuron.

Finally, we have shown how to use mathematical model to build networks of spiking neurons. We used Izhikevich model in MATLAB code for recognizing English uppercase alphabet letters. As a result we observed reactions of neurons during simulated time steps. Also, as a result we have got the recognized image by the network.

8 Summary

The main purpose of this paperwork was the assignment according to which implementation of spiking neural networks and learning algorithms in Matlab had to be done. Application of spiking neural networks had to be for image recognition. The text mostly contains the survey about spiking neural networks. According to studies and research done, there are some important and famous models introduced in the work.

The text is devided into eight chapters. First two chapters are the introduction and the brief history of neural networks. The third chapter and its subchapters introduce existing famous models of neural network. The fourth and the fifth sections present some neural coding techniques and learning rules. The main work of the experiment is introduced in the sixth chapter, where the image rocognition with spiking neural network is explained in detail and the rsults are shown. Finally, the conclusion is made and presented.

According to the paper we understand the differencies between biological and artificial neurons. Studies of artificial intellignce already were started from the beginning of the 20th century. Spiking neural network (SNN) is the most recent and biologically accurate, the third generation of neural networks. The main idea of SNN is that neurons don't fire at each propagation cycle, but they fire when a membrane potential, membrane electric charge reaches a specific value. The information is transferred from one input neuron to another through the electric pulses/spikes.

The first model of neural network was very sipmle and the neurons used for that model were called perceptrons. Later so called feed forward neural network was developed and the latest version of artificial neural network is spiking neural network. Izhikevich developed the simple model of spiking neurons combining Hodgkin–Huxley type model and Integrate-and-fire (IF) models. It is used for image recognition problems, what we've seen in the main part of the work.

For the experiment of this paperwork English alphabet, uppercase letters were used and Izhikevich model. The model consisted of the system of two simple differential equations and we've seen the networks behavior and neurons reactions for each image. By changing or modifying some small parameters in the code we could see different results and reactions. The main result was the recognized alphabet letter.

This work was mostly about research and observations. For the future the code presented in the work can be modified and improved for better recognition.

References

[1] Daqi Liu and Shigang Yue, "Spiking Neural Network for Visual Pattern Recognition," University of Lincoln, Lincoln.

[2] Shruti R Kulkarni and Maryam Shojaei Baghini, "Spiking Neural Network based ASIC for Character Recognition," Ninth International Conference on Natural Computation (ICNC), Mumbai, 2013.

[3] Francois Christophe, Tommi Mikkonen, Vafa Andalibi, Kai Koskimies and Teemu Laukkarinen, "Pattern recognition with Spiking Neural Networks: a simple training method," Tampere University of Technology, Tampere, Finland.

[4] Kshitij Dhoble, Nuttapod Nuntalid, Giacomo Indiveri and Nikola Kasabov, "Online Spatio-Temporal Pattern Recognition with Evolving Spiking Neural Networks utilising Address Event Representation, Rank Order, and Temporal Spike Learning," World Congress on Computational Intelligence, Brisbane, Australia, 2012.

[5] N. Kasabov, "Evolving Spiking Neural Networks and Neurogenetic Systems for Spatio- and Spectro-Temporal Data Modelling and Pattern Recognition," IEEE WCCI, Springer-Verlag Berlin Heidelberg, 2012.

[6] Taras Iakymchuk, Alfredo Rosado-Munoz, Juan F Guerrero-Martinez, Manuel Bataller-Mompean and Jose V Frances-Villora, "Simplified spiking neural network architecture and STDP learning algorithm applied to image classification," EURASIP Journal on Image and Video Processing, Valencia, Spain, 2015.

[7] Simei Gomes Wysoski, Lubica Benuskova and Nikola Kasabov, "Adaptive Spiking Neural Networks for Audiovisual Pattern Recognition," Springer-Verlag Berlin Heidelberg, Auckland, New Zealand, 2008.

[8] Zhenmin Zhang, Qingxiang Wu, Zhiqiang Zhuo, Xiaowei Wang and Liuping Huang, "Wavelet transform and texture recognition based on spiking neural network for visual images," Elsevier, Fuzhou, China, 2014.

[9] Nikola Kasabov, Kshitij Dhoblea, Nuttapod Nuntalid and Giacomo Indiveri, "Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition," Elsevier, Zurich, Switzerland, 2013.

[10] Joo-Heon Shin, David Smith, Waldemar Swiercz, Kevin Staley, J. Terry Rickard, Javier Montero, Lukasz A. Kurgan and Krzysztof J. Cios, "Recognition of Partially Occluded and Rotated Images With a Network of Spiking Neurons," IEEE TRANSACTIONS ON NEURAL NETWORKS, 2010.

[11] Mrs.Soni Chaturvedi, Dr. A.A. Khurshid and Dr. S.S. Dorle, "Reconfiguration of Spiking Neural Network for Optimization with Applications to Image Processing," Sixth International Conference on Emerging Trends in Engineering and Technology, Nagpur, 2013.

[12] J.L. Rossello, V. Canals, A. Oliver, M. Alomar and A. Morro, "SPIKING NEURAL NETWORKS SIGNAL PROCESSING," IEEE, Palma, Majorca, 2014.

[13] D.T. Pham, M.S. Packianather and E.Y.A. Charles, "Spiking neural network for control chart pattern recognition," Proceedings of the European Control Conference, Kos, Greece, 2007.

[14] Arash Ahmadi and Mark Zwolinski, "A Modified Izhikevich Model For Circuit Implementation of Spiking Neural Networks," University of Southampton, Southampton, 2010.

[15] E. Y. A. Charles, "An Efficient Discrete Model for Implementing Temporal Coding Spiking Neural Network," International Conference on Advances in ICT for Emerging Regions, Thirunelveli, Sri Lanka, 2014.

[16] S N Sivanandam, S Sumathi and S N Deepa, "Comparison between the brain and the computer, Comparison between artificial and biological neural network," in *Introduction to neural networks using MATLAB 6.0*, New Delhi, McGraw-Hill Offices, 2006, p. 17.

[17] J. Vreeken, "Spikin neural networks, an introduction," Utrecht.

[18] Mohammad A. Bhuiyan, Rommel Jalasutram and Tarek M. Taha, "Character recognition with two spiking neural network models on multicore architectures," IEEE, Clemson, 2009.

[19] Soni Chaturvedi, Neha R. Sondhiya and Rutika N.Titre, "Izhikevich Model Based Pattern Classifier For Hand Written Character Recognition - A Review Analysis," International Conference on Electronic Systems, Signal Processing and Computing Technologies, Nagpur, 2014.

[20] K. Dhoble, "Spatio-/Spectro-Temporal Pattern Recognition using Evolving Probabilistic Spiking Neural Networks," The Auckland University of Technology (PHD Thesis), Auckland, 2013.

[21] Kiyoshi Kawaguchi, Bsee, "A multithreaded software model for backpropagation neural network applications," The University of Texas at El Paso (Thesis), Texas, 2000.

[22] S. Donachy, "Spiking Neural Networks: Neuron Models, Plasticity, and Graph Applications," Virginia Commonwealth University, Virginia, 2015.

[23] E. Orhan, "The Leaky Integrate-and-Fire Neuron Model," 2012.

[24] Qiang Yu, K.C. Tan and Huajin Tang, "Pattern Recognition Computation in A Spiking Neural Network with Temporal Encoding and Learning," IEEE World Congress on Computational Intelligence, Brisbane, 2012.

[25] Jinling Wang, Ammar Belatreche, Liam Maguire and T. M. McGinnity, "Dynamically Evolving Spiking Neural Network for Pattern Recognition," IEEE, UK, 2015.

[26] Evangelos Stromatias and John S. Marsland, "Supervised learning in Spiking Neural Networks with Limited Precision: SNN/LP," IEEE, Manchester, Liverpool, UK, 2015.

[27] Long Peng, Zeng-Guang Hou, Nikola Kasabov, Gui-Bin Bian, Luige Vladareanu and Hongnian Yu, "Feasibility of NeuCube Spiking Neural Network Architecture for EMG Pattern Recognition," International Conference on Advanced Mechatronic Systems, Beijing, China, 2015.

[28] Xiurui Xie, Hong Qu, Zhang Yi and Jürgen Kurths, "Efficient Training of Supervised Spiking Neural Network via Accurate Synaptic-Efficiency Adjustment Method," IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, 2016.

[29] Yevgeniy Bodyanskiy, Artem Dolotov, Iryna Pliss and Mykola Malyar, "A Fast Learning Algorithm of Self-Learning Spiking Neural Network," IEEE First International Conference on Data Stream Mining & Processing, Lviv, Ukraine, 2016.

[30] Elahe Eskandari, Arash Ahmadi, Shaghayegh Gomar, Majid Ahmadi and Mehrdad Saif, "Evolving Spiking Neural Networks of Artificial Creatures Using Genetic Algorithm," IEEE, Ontario, Canada, 2016.

[31] Anne Abbott, Neelava Sengupta and Nikola Kasabov, "Which Method to Use for Optimal Structure and Function Representation of Large Spiking Neural Networks: A Case Study on the NeuCube Architecture," IEEE, Auckland, New Zealand, 2016.

[32] Khadeer Ahmed, Amar Shrestha, Qinru Qiu and Qing Wu, "Probabilistic Inference Using Stochastic Spiking Neural Networks on A Neurosynaptic Processor," IEEE, NY, USA, 2016.

[33] Nikola Kasabov, Lei Zhou, Maryam G. Doborjeh, Zohreh Gholami and Jie Yang, "New Algorithms for Encoding, Learning and Classification of fMRI Data in a Spiking Neural Network Architecture: A Case on Modelling and Understanding of Dynamic Cognitive Processes," IEEE Transaction on Cognitive and Developmental Systems, Auckland, New Zeland; Shanghai, China, 2016.

[34] Aleksandr A. Maliavko and Andrey V. Gavrilov, "Towards Development of Self-Learning and Self-Modification Spiking Neural Network as Model of Brain," 13th International Scientific-Technical Conference APEIE, Novosibirsk, Russia, 2016.

[35] E. M. Izhikevich, "Simple Model of Spiking Neurons," IEEE TRANSACTIONS ON NEURAL NETWORKS, 2003.

[36] N. Nuntalid, "Evolving Probabilistic Spiking Neural Networks for Modelling and Pattern Recognition of Spatio-temporal Data on the Case Study of Electroencephalography (EEG) Brain Data," PhD Thesis, Auckland, New Zeland, 2012.

[37] PRTools, "A Matlab Toolbox for Pattern Recognition," Pattern Recognition Group, The Netherlands, 2000.

[38] "The Integrate-and-Fire Model".

[39] IVAN BOGDANOV, RADU MIRSU and VIRGIL TIPONUT, "MATLAB MODEL FOR SPIKING NEURAL NETWORKS," Proceedings of the 13th WSEAS International Conference on SYSTEMS, Timisoara, Romania.

[40] Tianqi Tang, Lixue Xia, Boxun Li, Rong Luo, Yiran Chen, Yu Wang and Huazhong Yang, "Spiking Neural Network with RRAM: Can We Use It for Real-World Application?," EDAA, Beijing, China; Pittsburgh, USA, 2015.

Appendix 1 – SNN Code

```
temp = imread('letterA1.jpg'); temp = double(temp); % Convert the
image file onto numeric array
input_image = temp(:,:,1); clear temp % Load only one matrix entry in
case of color images

%% Parameters
int_time_step_v_EXC =0.2500;   % integration rate of variable v
int_time_step_u_EXC = 0.0200;  % integration rate of variable u

% coefficient that multiplies the input RGB value (in this case, it is
a single value between 0 and 255)
inp_ratio = 0.2500;

resting_potential = -65;  % the resting potential of the cell
simulation_length = 200;  % the simulated time steps

%% Size population
N_Rec = size(input_image);  % Receptors; total # of receptor neurons

%% Receptors variables
Rec_record =
resting_potential*ones(N_Rec(1)*N_Rec(2),simulation_length);
%Initialize Receptors' cell voltages
Rec_v = resting_potential*ones(N_Rec);  % Initialise Receptors'
membrane potential
Rec_u = .2*Rec_v;                       % Initialise Receptors'
recovery variable

% Initialize figure for plots
h = figure; set(h, 'DoubleBuffer', 'on', 'color', 'white', 'Name',
'Reduced spiking equation demo');

%% Simulate for t time steps
for t = 1:simulation_length; %there will be simulation_length # of
time steps

    %%%%%%%%%%%%% Receptors %%%%%%%%%%%%%
    Rec_fired = find(Rec_v >= 30);             % Search the indices of
receptor neurons for spikes by seeing whether it has crossed the
threshohold of 30mV
    if ~isempty(Rec_fired)                     % Reset F1 neurons
after firing if Rec_v has crossed the spiking threshhold
        Rec_v(Rec_fired) = resting_potential;  % The receptor's
membrane potencial is reset to its resting potencial
        Rec_u(Rec_fired) = Rec_u(Rec_fired)+ 8;  % The receptor's
recovery variable is increased by a parameter value equal to 8
    end

    % Perform numerical integration using Euler's method
```

```
    Rec_v = Rec_v + int_time_step_v_EXC*((0.04*Rec_v + 5) .* Rec_v +
140 - Rec_u + input_image * inp_ratio);
    Rec_u = Rec_u + int_time_step_u_EXC .*(0.2 * Rec_v - Rec_u);

    % Store Receptors variables
    Rec_record(:,t) = reshape(Rec_v, 1, N_Rec(1)*N_Rec(2));
    Rec_record(Rec_record >= 30)=30; % Normalize spikes: in real
neurons they cannot be above 30mV

    % Create PLOT

    subplot(2,2,1), imagesc(input_image), title(['Input image, cycle
', num2str(t), ' of ', num2str(simulation_length)]), axis off,...
        colormap gray, drawnow % display the loaded image and t time
step
    subplot(2,2,2), imagesc(Rec_v>30, [0 1]), drawnow,
title('Receptor spikes'), colormap gray, drawnow % Display the neurion
that are spiking at time t

    subplot(2,2,3), drawnow, plot(Rec_record(10,1:t), 'linewidth', 1),
xlim([0 200]), ylim([-75 40]), title('Voltage of receptor #10') %Plot
the voltage change of receptor #10
    subplot(2,2,4), drawnow, plot(Rec_record(400,1:t), 'red',
'linewidth',1), xlim([0 200]), ylim([-75 40]), title('Voltage of
receptor #200') %Plot the voltage change of receptor #200
end
```