Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Analysis of modulated spiking neural network

Bachelorarbeit
im Arbeitsbereich Knowledge Technology, WTM
Dr. Sven Magg

Department Informatik
MIN-Fakultät
Universität Hamburg

vorgelegt von
Marcus Soll
am
01.02.2016

Gutachter: Dr. Sven Magg
Dipl.-Inform. Stefan Heinrich

Marcus Soll
Matrikelnummer: 6427921
Max-Eichholz-Ring 45g
21031 Hamburg

Abstract

In his Bachelor thesis Stefan Bruhns has created a new network type called “modulated spiking neural network”. This thesis expands his work by analysing the performance of the new network on different tasks and comparing them to different other network types.

Zusammenfassung

In seiner Bachelorarbeit hat Stefan Bruhms einen neuen Typ neuronaler Netzerke namens “modulated spiking neural network” geschaffen. Diese Arbeit liefert eine vertiefende Analyse des neuen Netzwerkes, bei welcher das Netzwerk verschiedene Aufgaben lösen muss. Zus¨tzlich wird die Leistung der modulated spiking neural network mit der anderer Netzwerktypen verglichen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Neural Networks are widely used for a variety of tasks [WRL94] including tasks from clinical medicine [Bax95]. These networks are often adjusted through the process of evolution [SWE92]. Therefore it is necessary for a neural network to perform well in terms of evolvability.

In an attempt to find neural networks that perform better for certain tasks, or have improved characteristics (e.g. higher memorisation capabilities), new kinds of networks are constantly developed. Two examples are:

- GasNets [HSJO98, Hus98]. These networks are inspired by the discovery of freely floating nitric oxide in the brain. The neurons in this kind of networks have the ability to release different gases which modulate the behaviour of nearby neurons.

- Spiking neural networks [Maa97, BRC+07]. These kind of networks try to imitate the natural spiking behaviour of neurons instead of using an artificial activation function.

In his bachelor thesis Stefan Bruhns [Bru15] has combined these two network types to create a new type of neural network called "modulated spiking neural network". He has also performed some testing on the performance of this network type in terms of evolvability which has shown some promising results. His analysis was done on a single task though, so it can not be generalised in terms of complexity and type of task.

In order to get a better understanding of how well these modulated spiking neural work in different conditions and to get a better understanding of the characteristics of this new network type, I did some more empirical study on different simulations with different attributes (e.g. some where pattern generators or memory are useful and some where they are not) in my bachelor thesis. This way I hope to find out more about the advantages and disadvantages of the modulated spiking neural network architecture. One focus will lie on the usage of gas in the networks because the usage of gas has already shown a great impact for GasNets [Hus98]. The assumption is that the release of gas in combination with the spiking neurons will lead to a powerful network architecture.

Chapter 2

Theoretical background on used neural network models

The development of artificial neural networks (ANN) is inspired by the computational power of the human brain. The human brain is a complex biological system which can process information nonlinear and parallel [Kan11]. To build systems with corresponding features, multiple models have been developed which share a common structure:

Artificial neural networks consist of multiple processing units [KvdS96] (also called nodes or neurons [Kan11]). These processing units (See figure 2.1) consist of multiple input connections. The input value of this connection $(x_1, x_2, ..., x_m)$ is multiplied by a weight specific to the connection $(w_{k1}, w_{k2}, ..., w_{km})$. The sum of these connections (net) is then processed by an activation function $(f(net))$. The output of this activation function is the output of the neuron. In some models an additional connection with a constant value is used to modify the default behaviour of the neurons. This constant value is called bias (b_k).

There are multiple functions which are commonly used as activation functions.

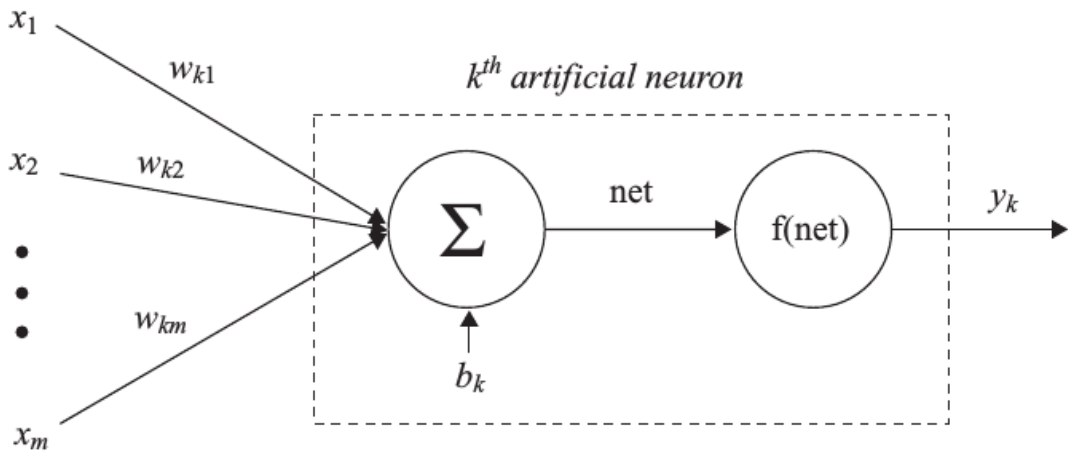


Figure 2.1: Simple model of an artificial neuron [Kan11]

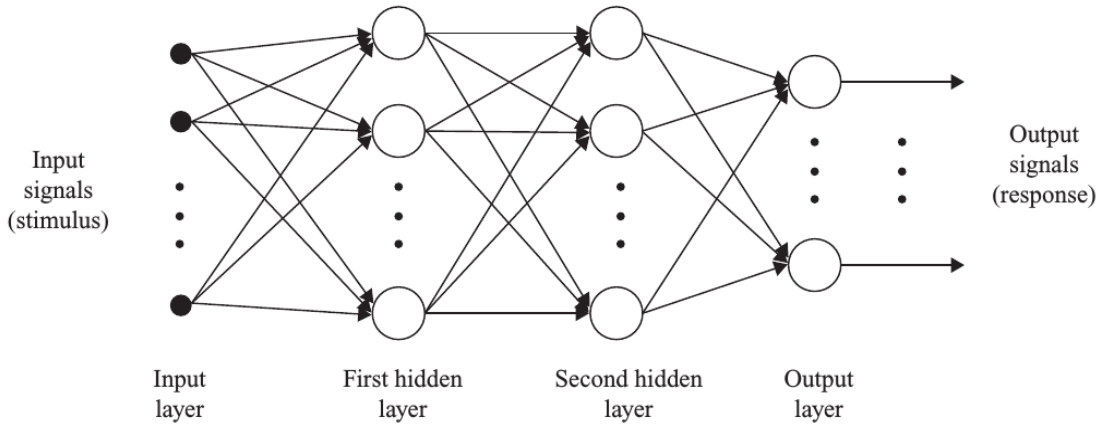


Figure 2.2: Schematic draft of a multi-layer feed-forward neural network with 2 hidden layers [Kan11]

These include the log-sigmoid (2.1) and hyperbolic tangent sigmoid (2.2) [Kan11].

$$y = \frac{1}{1 + e^{-net}} \tag{2.1}$$

$$y = \frac{e^{net} - e^{-net}}{e^{net} + e^{-net}} \tag{2.2}$$

The different network types used in this study can be distinguished by the way the neurons are connected and the specific way the neurons work internally.

2.1 Multi-layer feed-forward neural network

A feed-forward neural network (*FFN*) is a neural network where no feedback-connections are present, which means that the output of a neuron will be neither directly nor indirectly the input of the same neuron [KvdS96]. The network used in this thesis can be classified as a multi-layer feed-forward neural network (also called "multilayer perceptron"), which means that the neurons are clustered in layers. The input of each layer consists of the output of the prior layer (with the exception of the first layer whose input is from external sources). The output of the last layer is the output of the whole neural network. The first layer is called "input layer", the last layer is defined as "output layer". All other layers are called "hidden layer" (see figure 2.2).

Multi-layer feed-forward neural networks (together with a learning method called "backpropagation" as the learning algorithm) are probably the most commonly used neural network in the industry [Kan11]

2.2 Continuous-time recurrent neural network

A standard continuous-time recurrent neural network (*CTRNN*) [Ran95] is a neural network where all neurons have a connection to each other as well as a recurrent connection. The neurons in this type of network have an internal value y which is changing over time as well as a special output function depending on the internal value and the bias of the neuron. The output o_i^t and the internal value y_i^t at time t are defined as following:

$$o_i^t = \sigma(y_i^t + b_i) \tag{2.3}$$

$$y_i^t = y_i^{t-1} + \frac{\Delta t}{\tau_i}\left(-y_i^{t-1} + I_i^t + \sum_{j=i}^{N}(w_{ji} \cdot o_j^{t-1})\right) \tag{2.4}$$

where $\tau > 0$ is the time constant of the neuron, w_{ji} the weight of the connection from j to i, b_i the bias, I_i^t the input at time t and σ the activation function (which is often the sigmoid function as defined in (2.1)).

2.3 GasNet

The development of *GasNets* was inspired by the discovery of freely floating nitric oxide in the human brain [HSJO98]. The neurons in this type of network are arranged on a two-dimensional plane. Each neuron has the ability to release one of two different simulated gases (called gas1 and gas2) which can influence the activation function (2.5) of other neurons by changing the parameter K.

$$y_i^t = \tanh\left[K_i^t\left(I_i^t + \sum_{j=1}^{N}(w_{ji} \cdot y_j^{t-1})\right) + b_i\right] \tag{2.5}$$

where tanh is the hyperbolic tangent function (2.2), I_i^t the input at time t, $w_j i$ the weight of the connection from j to i and b_i the bias. K_i^t is the transfer function parameter which can be increased through gas1 and decreased through gas2 as following:

$$K_i^t = P[D_i^t] \tag{2.6}$$

$$P = \{-4.0, -2.0, -1.0, -0.5, -0.25, -0.125, 0.0, 0.125, 0.25, 0.5, 1.0, 2.0, 4.0\} \tag{2.7}$$

$$D_i^t = f\left(D_i^0 + C_1^t \cdot (13 - D_i^0) - C_2^t \cdot D_i^0\right) \tag{2.8}$$

$$f(x) = \begin{cases} 0 & x \leq 0 \\ \lfloor x \rfloor & 0 < x < N \\ N & else \end{cases} \tag{2.9}$$

where D_i^0 is the default index specific to a neuron, C_1^t is the concentration of gas1 at time t and C_1^t is the concentration of gas2 at time t. Both are calculated through (2.10) [HSJO98, Hus98].

$$C(d,t) = \begin{cases} C_0 \cdot e^{\frac{-2d}{r}} \cdot T_t(t) & d < r \\ 0 & else \end{cases} \tag{2.10}$$

$$T_t(t) = \begin{cases} H(T_{t-1}(t) + \frac{1}{k}) & \text{emitting} \\ H(T_{t-1}(t) - \frac{1}{k}) & \text{emitting} \end{cases} \tag{2.11}$$

$$H(x) = \begin{cases} 0 & x \leq 0 \\ x & 0 < x < 1 \\ 1 & else \end{cases} \tag{2.12}$$

where C_0 is a global constant (set to 1 for this study). Several recent publications used different functions to calculate the gas concentration [SHPO02, Smi02, MP06], see appendix A for more information.

A gas is released when either the electric charge exceeds a threshold (0.5), gas 1 concentration exceeds a threshold (0.1) or gas 2 exceeds a threshold (0.1). The type of gas emitted as well as the emitting condition is determined genetically.

The outgoing connections of a neuron are defined through two cones (also called segments): One positive segment and one negative segments. Each neuron in a positive section has a excitatory connection (with the weight of 1) from the parent neuron of the cone and each neuron in a negative sector a inhibitory connection (with the weight of -1) [HSJO98, Hus98]. One example can be seen in figure 2.3.

2.4 Modulated spiking neural network

While all neurons in the prior neural network types use an artificial activation function, spiking neurons try to emulate the electrical activity of real brain neurons. One of this models is the Izhikevich-model [Izh03], which can be described as following:

$$\Delta_t v = 0.04 v_t^2 + 5 v_t + 140 - u_t + I \tag{2.13}$$

$$\Delta_t u = a(b \cdot v - u) \tag{2.14}$$

$$v_{t+1} = v_t + \Delta_t v \tag{2.15}$$

$$u_{t+1} = u_t + \Delta_t u \tag{2.16}$$

where v modulates the electric charge of of the neuron and u decribes the behaviour after a spike. The neuron resets after a spike as following:

$$\text{if } v \geq 30, \text{ then } \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{2.17}$$

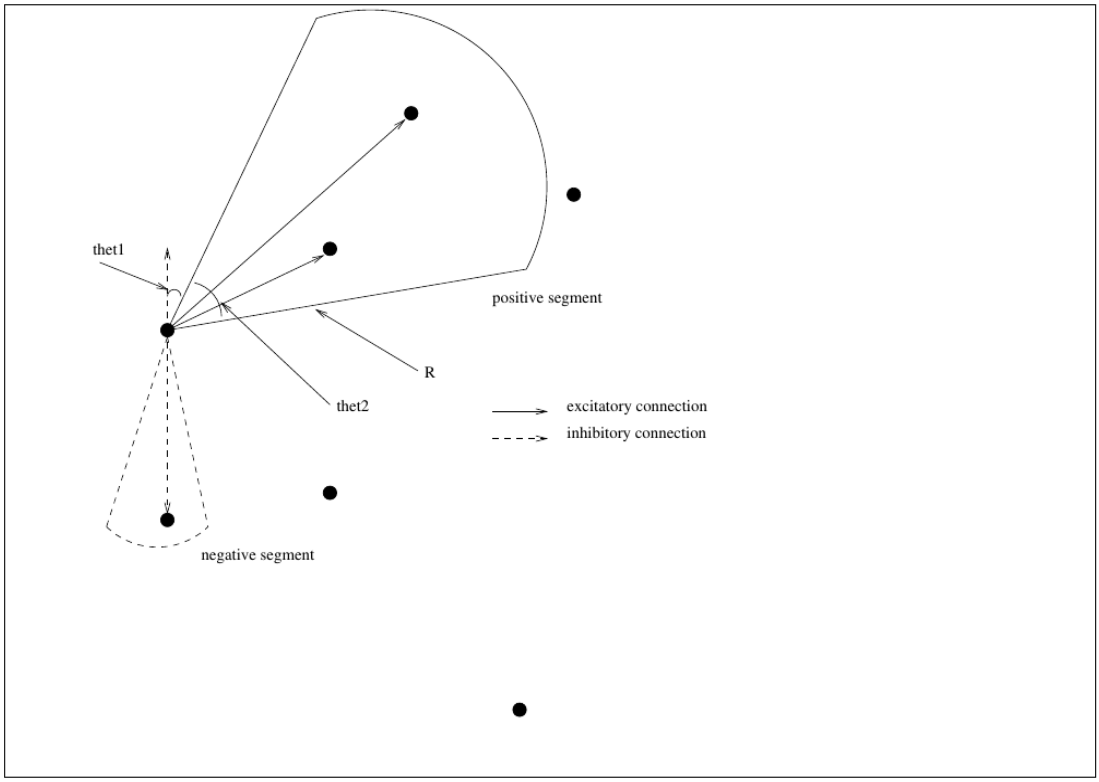


Figure 2.3: Connection scheme in a GasNet [Hus98]

a, b, c, d are variables which describe the behaviour of the network.

In his bachelor thesis Stefan Bruhns combines this spiking neuron model with the basic architecture of GasNets to create a new neural network type called "modulated spiking neural network" (*MSNN*) or "Modulated-SpikingNet" [Bru15]. The Izhikevich-model was chosen because it can model a lot of relevant neuron behaviours while at the same time being reasonably efficient [Izh04]. Stefan Bruhns has chosen the following sets for the parameters [Bru15], unfortunately he does not give a reason why he has chosen them:

$$P_a = \{0.0, 0.02, 0.04, 0.045, 0.0475, 0.04875, 0.05, 0.05125, 0.0525, 0.055, 0.06, 0.08, 0.1\} \tag{2.18}$$

$$P_b = \{-0.2, 0.0, 0.1, 0.15, 0.175, 0.1875, 0.2, 0.2125, 0.225, 0.25, 0.3, 0.4, 0.6\} \tag{2.19}$$

$$P_c = \{-80, -72, -68, -66, -65.5, -65.25, -65, -64.75, -64.5, -64, -62, -58, -50\} \tag{2.20}$$

$$P_d = \{-2, 0, 1, 1.5, 1.75, 1.875, 2, 2.125, 2.25, 2.5, 3, 4, 6\} \tag{2.21}$$

These variables can either be fixed for each neuron or modulated through two gases

like GasNets (see section 2.3). The values are modulated as following:

$$X_i^t = P_x[Index_{X,i}^t] \tag{2.22}$$

$$Index_{X,i}^t = f\left(Index_X^0 + C_{X,positiv}^t \cdot (13 - D_{X,i}^0) - C_{X,negativ}^t \cdot D_{X,i}^0\right) \tag{2.23}$$

$$f(x) = \begin{cases} 0 & x \leq 0 \\ \lfloor x \rfloor & 0 < x < N \\ N & else \end{cases} \tag{2.24}$$

where X is the variable which is modulated (a, b, c, d), $Index_{X,i}^0$ is the default index of variable X specific to neuron i, $C_{X,positiv}^t$ the gas which positively influences variable X and $C_{X,negativ}^t$ the gas which negatively influences variable X. Both $C_{X,positiv}^t$ and $C_{X,negativ}^t$ can be calculated through (2.10). Although this function differs from the one used by Stefan Bruhns [Bru15] it was chosen to make the results more comparable to the GasNets. The function itself should only have a minimal impact on the results of the modulated spiking neural networks (see appendix A).

A neuron emits gas when either the electric charge of the neuron exceeds a threshold (0.5) or the gas concentration of a genetic determined gas at the neuron exceeds a threshold (0.1). The type of gas emitted is determined genetically from a pool positive and negative gas of all modulated values.

The network calculates multiple steps internally for every simulation step (in this work 10 internal steps per simulation step). The output of a neuron equals to its spiking frequency over these internal steps:

$$o_i^t = \frac{N_{spikes,i}^t}{N_{internal\ steps}} \tag{2.25}$$

where o_i^t is the output of neuron i at simulation time step t, $N_{spikes,i}^t$ is the number of spikes of neuron i at simulation time step t and $N_{internal\ steps}$ is the number of internal time steps per simulation time step.

The connections in this network follow the same scheme as in the GasNets (see figure 2.3).

Chapter 3

Experimental setup

The following chapter describes the setup of the experiments. Each experiment consist of a simulation with a task for which each neural network has to find the best behaviour. This is done using a genetic algorithm.

3.1 Genetic Algorithm

Genetic algorithms are often used to get near optimal solutions for an optimisation problem [Kan11]. A genetic algorithm works on a population of possible solutions, where each individual has a fitness which describes how good the solution is for the given problem. Each individual is encoded into a chromosome containing all important information. In this study a genetic algorithm is used to optimise the structure of the neural networks. The process of a genetic algorithm is shown in figure 3.2, which contains the following steps [Cox05]:

1. The individuals must be encoded into a chromosome. In this study the chromosome contains several genes. All genes have the same size (which depends on the type of network that is encoded, see section 3.3) and contains positive integers in the range $[0, 2147483647]$ called allele (which is the positive range of a 32-bit integer variable). This alleles can then be transformed in numbers of different ranges as needed by the networks. The exact number of genes in a chromosome is dependent on the type of network (see section 3.3). The fitness function is given by the simulation.

2. The population must be initialised with individuals. For this first population all chromosomes are created with random values.

3. Each individual has to be evaluated to determine its fitness, therefore the chromosome is transformed into the corresponding network which is evaluated through the simulation's fitness function. For individuals which were already present in the last generation we do not need to recalculate its fitness.

4. After the evaluation we can determine if we want to terminate the genetic algorithm. There are a variety of termination reasons including a certain

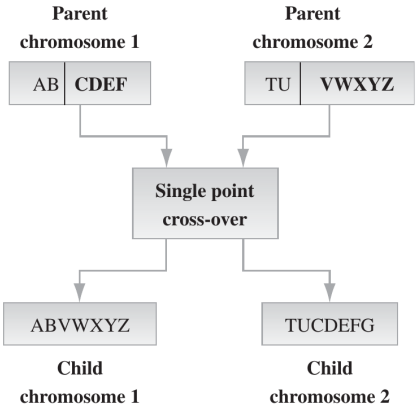


Figure 3.1: One-point crossover [Cox05]

fitness value has been exceeded by an individual or a certain amount of rounds has been reached.

5. Finally a new population has to be build. The steps of building the new population are defined as following:

   - **Creating children:** First a section of the the individuals are combined to create new offspring (also called children). In this study the whole population is divided into multiple tournament groups containing 8 individuals (the last group may be smaller when the population size is not dividable by 8 without a remainder). In every group the best two individuals are then combined using a one-point crossover / single-point crossover (see figure 3.1) and the new children are added to the tournament group.
   - **Mutation:** To overcome local maxima it is important to add some mutation to individuals. In this study each of the children gets mutated where each allele can be mutated by a small chance (3%). If an allele is mutated, its value get changed to a random number. Additionally there are some chromosome types where the length is variable in the range $[s, 4 \cdot s]$, where s is the minimal amount of output neurons required by the simulation. In this case, there is a certain chance (3%) that a gene is deleted or a random gene is added (both can occur at the same time).
   - **Survivor selection:** After the creation of children the population has to be reduced to its original size. Therefore, a number of survivors (equal to the original population size) are selected. In this study, the best 8 individuals out of a tournament group are selected for the new population.

The size of the population in this study is 300. The genetic algorithm runs until 200 rounds have passed or the fitness of the best individual exceeds 0.99.

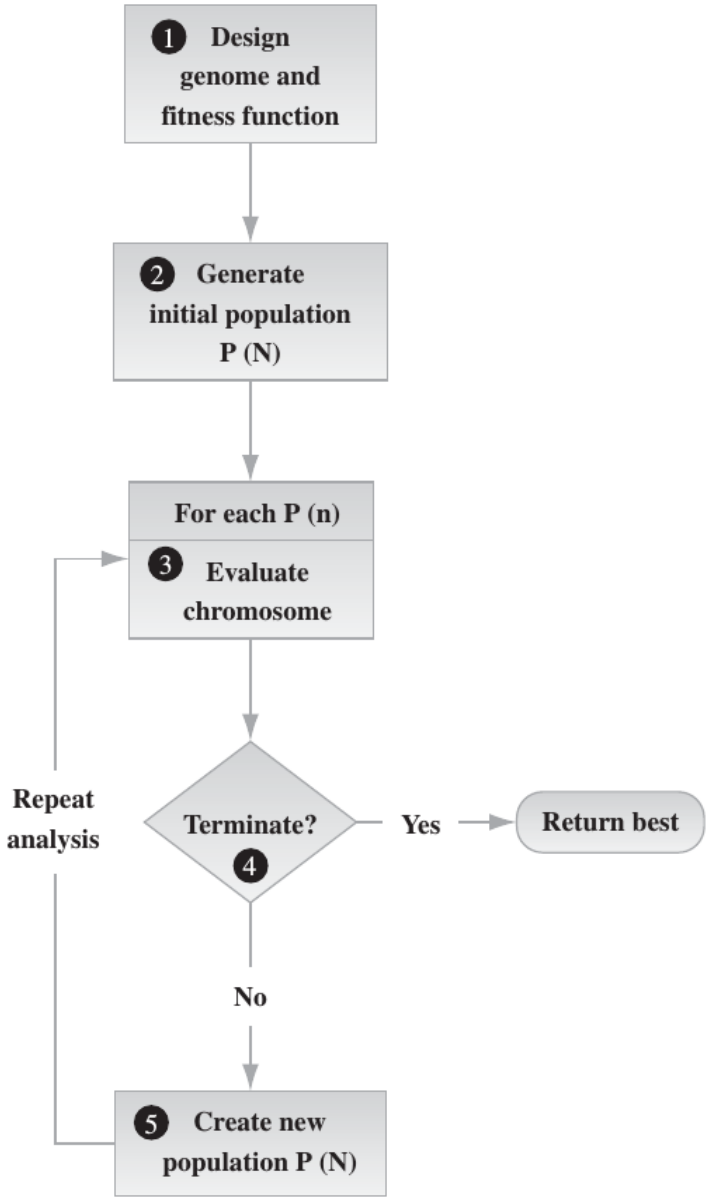


Figure 3.2: Basic process of a genetic algorithm [Cox05]

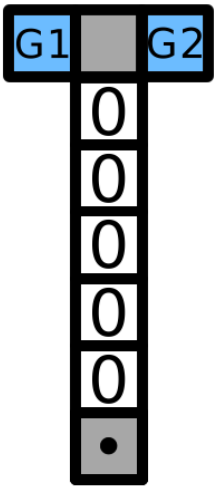


Figure 3.3: Example of a normal t-maze. Grey tiles represent changeable tiles, blue areas represent goals. The dot is the start position.

3.2 Simulations

Each simulation is a task which a neural network has to solve. All simulations have the same basic structure: The simulation is repeated N_{trials} number of times which is specific to the type of simulation. All runs are randomised. This has the consequence that the networks have to find a way to solve arbitrary variants of the simulation instead of learning a fixed sequence of steps. The fitness can be calculated as following:

$$fitness = \frac{N_{successful}}{N_{trials}} \tag{3.1}$$

where $N_{successful}$ is the number of times the simulation was completed successfully.

The size of the input and the number of required output neurons is completely dependent on the type of simulation.

3.2.1 T-Maze

The t-maze simulation used in this thesis is an abstract simulation. A robot has to walk down a corridor consisting of tiles and at the end has to choose between two goals. At the beginning and at the end (grey tiles in image 3.3) are two random numbers between one and five. If both numbers are the same the robot has to go to the goal **"g1"**, else the correct goal is **"g2"** (see image 3.3). The simulation is considered successful if the robot chooses the correct goal in a given time limit. Both goals are correct with a chance of 50%. This task was chosen because it is an easy task where a neural network has to have memory to find the correct goal.

The network has five input values representing the five possible numbers. If the

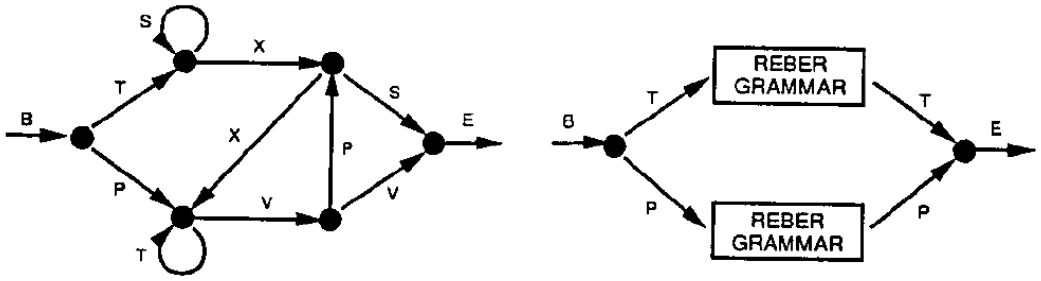


Figure 3.4: State machine of Reber grammar (left) and embedded Reber grammar (right) [Fah91]

number is not null the value representing the number will be set to 1. All other numbers are set to 0.

The task requires four output neurons, each representing one direction (left, right, up, down). The output with the highest value will be the direction in which the robot goes one tile, if there are multiple highest values or there is no tile in the chosen direction the robot does not move. The simulation is repeated 250 times ($N_{trials} = 250$).

There are two variants of the t-maze:

- **Normal t-maze**: This maze contains 5 tiles with 0-values (7 in total) and must be completed within 50 time steps. It is identical to the t-maze in figure 3.3. To solve this maze, short-term memory should be sufficient to choose the correct goal.

- **Huge t-maze**: This maze contains 150 tiles with 0-values (152 in total) and must be completed within 500 time steps. Because of the high amount of tiles between the start and the end the network requires long-term memory to choose the correct goal.

3.2.2 Reber grammar

The Reber grammars are a set of grammars based on a finite state machine originally developed by Reber [Reb67]. To learn the grammar, a neural network must generalise the words in the presence of noise as well as memory for the past states of the network [Fah91]. A special form is the "embedded Reber grammar", where the network has to memorise the beginning of the word in order to assign the correct ending of the word. The used grammars can be seen in figure 3.4. In this grammar, a word can consist of the letters `BTSXEPV`. The input consists of 7 values, each representing one character, where the value for the current character is set to 1 and all other values are set to 0.

The task can be split in two parts. Each part was performed with both grammars:

- **Detect a word**: A word which consists of the letters `BTSXEPV` is inputted into the network character by character. This word is either a valid word or a word where some letters have been replaced to get an invalid word. After all characters have been processed the network has to output one value. If the value exceeds a threshold (0.5) the network accepts the word, else the network rejects it. A trial is successful if a valid word is accepted or an invalid word is rejected. This is repeated 500 times ($N_{trials} = 500$).

- **Create a word**: The beginning of a word is inputted into a neural network (half to three quarters of a word), after this the network has to finish the word. The network has eight output values, seven of them represent the letters `BTSXEPV` and one indicates the termination of a word. The highest value is assumed to be the output value which is then used as an input for the next step. A word has to end with the termination output and may not be larger then a given length (50). If the word is a valid word the trial is successful. This is repeated 50 times ($N_{trials} = 50$).

3.3 Networks

The following section describes the concrete specification of the networks including the encoding of the genes (see section 3.1). Each number is transformed from the corresponding allele (range $[0, 2147483647]$) to a value in the range needed by the value. The theoretical background of the networks can be found in chapter 2.

3.3.1 Feed-forward neural network

The feed-forward neural network (*FFN*) used in this thesis is a multi-layer perceptron (section 2.1) containing 2 hidden layer with 5 neurons each. Each neuron (except the input neurons) has a special bias connection which has a constant input of 1. The size of the input and the output layer depends on the requirements of the simulation used. The encoding is as following:

$$< chromosome > := (< gene >)^n$$
$$< gene > :=< weight >$$

- n: Number of connections.

- $weight$: Weight of nth connection ($[-1, 1]$).

3.3.2 Continuous-time recurrent neural network

The chromosome of a continuous-time recurrent neural network (*CTRNN*) is composed as following, where a gene represents a neuron:

$$< chromosome > := (< gene >)^n$$
$$< gene > :=< b >< I >< \tau >< W >$$

- n: Number of neurons.
- b: Bias of the neuron ($[-5, 5]$).
- I: External input of the neuron. Either the number of the input or no input.
- τ: Time constraint of the neuron ($[1, 5]$).
- W: List of weights for all incoming connections (Range weight: $[-5, 5]$).

There are different versions tested in this thesis. The activation function can either be the sigmoid function (2.1) or the tanh function (2.2). The size of the network is either seven neurons large (14 in case of Reber "create a word", section 3.2.2) or the size is changing in the process of evolution.

3.3.3 GasNet

The chromosome of a *GasNet* is composed as following, where a gene represents a neuron. The same or an equal encoding with different input encoding was used by different prior work on GasNets [HSJO98, Hus98, Bru15, Smi02]:

$$< chromosome >:=(< gene >)^n$$
$$< gene >:= < x >< y >< R_p >< \Theta_{1p} >< \Theta_{2p} >< R_n >< \Theta_{1n} >$$
$$< \Theta_{2n} >< I >< rec >< TE >< CE >< s >< R^e >$$
$$< D^0 >< b >$$

- n: Number of neurons.
- x: Horizontal position of the neuron ($[0, 1]$).
- y: Vertical position of the neuron ($[0, 1]$).
- R_p: Radius of the positive cone ($[0\%, 50\%]$ of the area).
- Θ_{1p}: Angular extend of the positive cone ($[0, 2\pi]$).
- Θ_{2p}: Orientation of positive cone ($[0, 2\pi]$).
- R_n: Radius of the negative cone ($[0\%, 50\%]$ of the area).
- Θ_{1n}: Angular extend of the negative cone ($[0, 2\pi]$).
- Θ_{2n}: Orientation of negative cone ($[0, 2\pi]$).

- I: External input of the neuron. Either the number of the input or no input.
- rec: Weight of recurrent connection ($\{0, 1, -1\}$).
- TE: Circumstances of gas output. (0 = no gas, 1 = activation > 0.5, 2 = gas1 concentration > 0.1, 3 = gas2 concentration > 0.1).
- CE: Type of gas that is emitted ($\{$No gas, gas1, gas2$\}$).
- s: Rate of gas build up/decay ($[1, 11]$).
- R^e: Radius of emitted gas ($[10\%, 60\%]$ of the area).
- D^0: Value for D^0 in formula (2.8) ($\{1, 2, ..., 12, 13\}$).
- b: Bias of the neuron ($[-1, 1]$).

The length of the chromosome is changing in the process of evolution.

3.3.4 Modulated spiking neural network

The chromosome of a modulated spiking neural network (*MSNN*) is composed as following, where a gene represents a neuron. The encoding is the same as used by Stefan Bruhns [Bru15]:

$$< chromosome >:=(< gene >)^n$$
$$< gene >:= < x >< y >< R_p >< \Theta_{1p} >< \Theta_{2p} >< R_n >< \Theta_{1n} >$$
$$< \Theta_{2n} >< I >< rec >< TE >< CE >< s >< R^e >$$
$$< a >< b >< c >< d >$$

- n: Number of neurons.
- x: Horizontal position of the neuron ($[0, 1]$).
- y: Vertical position of the neuron ($[0, 1]$).
- R_p: Radius of the positive cone ($[0\%, 50\%]$ of the area).
- Θ_{1p}: Angular extend of the positive cone ($[0, 2\pi]$).
- Θ_{2p}: Orientation of positive cone ($[0, 2\pi]$).
- R_n: Radius of the negative cone ($[0\%, 50\%]$ of the area).
- Θ_{1n}: Angular extend of the negative cone ($[0, 2\pi]$).
- Θ_{2p}: Orientation of negative cone ($[0, 2\pi]$).
- I: External input of the neuron. Either the number of the input or no input.

- rec: Weight of recurrent connection ($\{0, 1, -1\}$).
- TE: Circumstances of gas output. ($\{$no gas, activation > 0.5, positive gas concentration > 0.1, negative gas concentration $> 0.1\}$).
- CE: Type of gas that is emitted ($\{$No gas, positive gas, neggative gas$\}$).
- s: Rate of gas build up/decay ($[1, 11]$).
- R^e: Radius of emitted gas ($[10\%, 60\%]$ of the area).
- a: Value for $Index_a^0$ in formula (2.23) ($\{1, 2, ..., 12, 13\}$).
- b: Value for $Index_b^0$ in formula (2.23) ($\{1, 2, ..., 12, 13\}$).
- c: Value for $Index_c^0$ in formula (2.23) ($\{1, 2, ..., 12, 13\}$).
- d: Value for $Index_d^0$ in formula (2.23) ($\{1, 2, ..., 12, 13\}$).

The length of the chromosome is changing in the process of evolution. The following variants have been tested:

- Variable a is modulated, all other variables are not modulated.
- Variable b is modulated, all other variables are not modulated.
- Variable c is modulated, all other variables are not modulated.
- Variable d is modulated, all other variables are not modulated.
- All variables are modulated.
- No variables are modulated.

For each modulated value a seperate positive and negative gas is emitted. Therefore the type of gases TE, CE depends on which values are modulated.

Chapter 4

Results

This chapter contains the results of the evolutionary processes. The acronyms used in this sections can be found in table 4.1. A detailed analysis of the modulated spiking neural networks can be found in chapter 5.

The aim of the experiments was to rank the modulated spiking neural networks among the other tested networks to get a better understanding of the performance of MSNN. Therefore the focus in this chapter lies more on the comparison of modulated spiking neural networks to other networks rather than a detailed analysis of the tasks.

The genetic algorithm was terminated when either the fitness of the best individual was higher than 0.99 or generation 200 was reached. Because the number of generation is equal across all tested neural networks the results should be comparable. In addition to that some testing with higher generation numbers did not show any significant improvement (see appendix B).

4.1 T-Maze

The results of the t-maze-simulation (see section 3.2.1) can be found in tables 4.2 - 4.3. We can see that only the modulated spiking neural networks (b-modulated in normal t-maze and d-modulated in huge t-maze) were capable of evolving networks in less than 200 generations which could solve the t-maze task with a success rate of over 99%. In general the modulated spiking neural networks seem to perform better than all other networks tested.

If you compare the MSNN where a single value is modulated with the MSNN without modulation you can see that the single modulated networks tend to perform better than the non modulated network and the fully modulated network, however the t-test does not show any significant difference ($P \geq 5\%$).

If you look at the continuous-time recurrent neural networks you can see that a fixed size of neurons does not seem to improve the results. However for the CTRNN the selection of the activation function seem to have an impact. Networks using the hyperbolic tangent function perform better in this task than the ones using the sigmoid function.

Network	Detailed description
FFN	multilayer perceptron with sigmoid function (2.1) as described in sections 2.1 and 3.3.1
FFN (tanh)	multilayer perceptron with sigmoid function (2.2) as described in sections 2.1 and 3.3.1
CTRNN	CTRNN with fixed size and sigmoid function (2.1) as described in sections 2.2 and 3.3.2
CTRNN (size changing)	CTRNN with changing size and sigmoid function (2.1) as described in sections 2.2 and 3.3.2
CTRNN (tanh)	CTRNN with fixed size and tanh function (2.2) as described in sections 2.2 and 3.3.2
CTRNN (tanh, size changing)	CTRNN with changing size and tanh function (2.2) as described in sections 2.2 and 3.3.2
GasNet	GasNet as as described in sections 2.3 and 3.3.3
MSNN (a)	Modulated spiking neural network (modulated vaues: a) as described in sections 2.4 and 3.3.4
MSNN (b)	Modulated spiking neural network (modulated vaues: b) as described in sections 2.4 and 3.3.4
MSNN (c)	Modulated spiking neural network (modulated vaues: c) as described in sections 2.4 and 3.3.4
MSNN (d)	Modulated spiking neural network (modulated vaues: d) as described in sections 2.4 and 3.3.4
MSNN (full)	Modulated spiking neural network (modulated vaues: a, b, c, d) as described in sections 2.4 and 3.3.4
MSNN (none)	Modulated spiking neural network (no modulated vaues) as described in sections 2.4 and 3.3.4

Table 4.1: Acronyms for networks

Network	Average best individual	Average fitness	Average number of generations
FFN	0.202 (0.010)	0.159 (0.003)	200.0 (0.0)
FFN (tanh)	0.200 (0.007)	0.160 (0.002)	200.0 (0.0)
CTRNN	0.770 (0.052)	0.705 (0.050)	200.0 (0.0)
CTRNN (size changing)	0.790 (0.066)	0.742 (0.067)	200.0 (0.0)
CTRNN (tanh)	0.855 (0.049)	0.755 (0.056)	200.0 (0.0)
CTRNN (tanh, size changing)	0.864 (0.028)	0.819 (0.030)	200.0 (0.0)
GasNet	0.835 (0.036)	0.785 (0.032)	200.0 (0.0)
MSNN (a)	0.898 (0.042)	0.857 (0.045)	200.0 (0.0)
MSNN (b)	0.902 (0.051)	0.862 (0.059)	194.5 (20.5)
MSNN (c)	0.909 (0.041)	0.865 (0.054)	200.0 (0.0)
MSNN (d)	0.891 (0.052)	0.846 (0.055)	200.0 (0.0)
MSNN (full)	0.880 (0.052)	0.841 (0.057)	200.0 (0.0)
MSNN (none)	0.882 (0.059)	0.836 (0.067)	200.0 (0.0)

Table 4.2: Average (standard deviation) of 25 evolutionary runs with standard t-maze simulation

The GasNet performed quiet well with the average best individual and average fitness being slightly lower than the ones of the CTRNN-tanh networks and higher than the CTRNN-sigmoid networks.

The feed-forward networks were successful in about one fifth of the trials. This shows that the task is not solvable for networks without memory.

Network	Average best individual	Average fitness	Average number of generations
FFN	0.199 (0.006)	0.158 (0.003)	200.0 (0.0)
FFN (tanh)	0.196 (0.006)	0.159 (0.002)	200.0 (0.0)
CTRNN	0.773 (0.058)	0.699 (0.042)	200.0 (0.0)
CTRNN (size changing)	0.781 (0.083)	0.731 (0.080)	200.0 (0.0)
CTRNN (tanh)	0.822 (0.040)	0.717 (0.048)	200.0 (0.0)
CTRNN (tanh, size changing)	0.847 (0.054)	0.799 (0.060)	200.0 (0.0)
GasNet	0.827 (0.036)	0.785 (0.036)	200.0 (0.0)
MSNN (a)	0.868 (0.072)	0.826 (0.078)	200.0 (0.0)
MSNN (b)	0.880 (0.067)	0.841 (0.071)	200.0 (0.0)
MSNN (c)	0.876 (0.058)	0.830 (0.069)	200.0 (0.0)
MSNN (d)	0.885 (0.072)	0.841 (0.078)	195.9 (20.6)
MSNN (full)	0.869 (0.068)	0.823 (0.071)	200.0 (0.0)
MSNN (none)	0.876 (0.068)	0.834 (0.071)	200.0 (0.0)

Table 4.3: Average (standard deviation) of 25 evolutionary runs with huge t-maze simulation

4.2 Reber grammar

The results of the t-maze-simulation (see section 3.2.2) can be found in tables 4.4 - 4.7.

The modulated spiking neural networks had problems to find solutions for this tasks. If we compare the average best individual or the average fitness of the MSNN with the values of the GasNet or the CTRNN variants we can see that some CTRNN-variants and the GasNet outperform the MSNN.

4.2.1 (embedded) Reber detect a word

In the "Reber detect a word" task the performance of the MSNN (independent of the modulated parameter) is around the performance of the feed-forward network (see table 4.4) while at the "embedded Reber detect a word" the performance is slightly better than the performance of the feed-forward network (see table 4.6). In both cases the performance difference between the different modulation variants is quite close. If you consider that the feed-forward networks can not solve the task properly because of missing memory, you can conclude that the MSNN was not able to learn how to detect reber grammar words. The same applies to all other network types.

If you compare the size changing continuous-time recurrent neural networks with the non size changing ones, you can see that having a fixed size of neurons seems to be an advantage in the "Reber detect a word" and the "embedded Reber detect a word" task. This is most probably due to the fact that the size changing networks tend to have a low number of neurons (often one or two) because it is easier to optimise one neuron than multiple. However at a certain point having more neurons allows the network to develop a more complex behaviour. At that point it is hard for the size changing networks to add additional neurons because

they do not improve the results over the few neurons directly.

One interesting note is that the GasNet performed as well as the non size changing CTRNNs. This is noteworthy because the GasNet itself is size changing.

To get some more information on this topic it would be good to run some more simulation with size changing and non size changing networks, however this lies beyond the scope of this bachelor thesis.

4.2.2 (embedded) Reber create a word

In the "Reber create a word" task (see table 4.5) the performance of the modulated spiking neural network is better or equal than the performance of most CTRNN variants (except size changing CTRNN with tanh function). However, the performance is way worse than the performance of the GasNet. This may indicate that some of the features which make the GasNet perform well on the "Reber create a word" task is missing in the MSNNs.

Generally speaking the GasNets have a high fitness compared to all other networks tested. This indicates that the structure of the GasNet is advantageous for this task.

One special exception for the "Reber create a word" task is the feed-forward network with hyperbolic tangent function, which found a deterministic way of correctly finishing most of the words. This is interesting not only because no other tested network type seem to manage this, but also because this could not be reproduced by the feed-forward networks using the sigmoid function as an activation function. It would be interesting to look deeper in the dynamics of the feed-forward networks for this task, however this would exceed the scope of this bachelor thesis.

The results for the "embedded Reber create a word" task (see table 4.7) are bad across all network types tested here. While the MSNN variants could outperform CTRNN and CTRNN size changing, it is worse than the performance of the CTRNN tanh variants and the GasNet.

Once again the GasNet outperformed all other tested network types. This further strengthens the theory that GasNets have some structural advantages over other network in this task.

Network	Average best individual	Average fitness	Average number of generations
FFN	0.640 (0.004)	0.609 (0.002)	200.0 (0.0)
FFN (tanh)	0.641 (0.005)	0.610 (0.002)	200.0 (0.0)
CTRNN	0.712 (0.033)	0.670 (0.032)	200.0 (0.0)
CTRNN (size changing)	0.652 (0.020)	0.624 (0.018)	200.0 (0.0)
CTRNN (tanh)	0.697 (0.033)	0.654 (0.029)	200.0 (0.0)
CTRNN (tanh, size changing)	0.673 (0.024)	0.642 (0.023)	200.0 (0.0)
GasNet	0.679 (0.022)	0.651 (0.020)	200.0 (0.0)
MSNN (a)	0.647 (0.029)	0.616 (0.028)	200.0 (0.0)
MSNN (b)	0.639 (0.029)	0.609 (0.026)	200.0 (0.0)
MSNN (c)	0.642 (0.034)	0.613 (0.031)	200.0 (0.0)
MSNN (d)	0.640 (0.039)	0.610 (0.035)	200.0 (0.0)
MSNN (full)	0.640 (0.038)	0.612 (0.036)	200.0 (0.0)
MSNN (none)	0.640 (0.035)	0.611 (0.033)	200.0 (0.0)

Table 4.4: Average (standard deviation) of 25 evolutionary runs with Reber grammar / detect a word

Network	Average best individual	Average fitness	Average number of generations
FFN	0.078 (0.201)	0.042 (0.156)	200.0 (0.0)
FFN (tanh)	0.889 (0.113)	0.736 (0.143)	200.0 (0.0)
CTRNN	0.211 (0.164)	0.075 (0.092)	200.0 (0.0)
CTRNN (size changing)	0.243 (0.195)	0.127 (0.160)	200.0 (0.0)
CTRNN (tanh)	0.344 (0.051)	0.186 (0.037)	200.0 (0.0)
CTRNN (tanh, size changing)	0.446 (0.050)	0.304 (0.058)	200.0 (0.0)
GasNet	0.722 (0.142)	0.585 (0.139)	193.76 (21.6)
MSNN (a)	0.342 (0.092)	0.193 (0.103)	200.0 (0.0)
MSNN (b)	0.343 (0.100)	0.202 (0.097)	200.0 (0.0)
MSNN (c)	0.326 (0.095)	0.184 (0.088)	200.0 (0.0)
MSNN (d)	0.332 (0.105)	0.183 (0.093)	200.0 (0.0)
MSNN (full)	0.281 (0.107)	0.146 (0.111)	200.0 (0.0)
MSNN (none)	0.356 (0.094)	0.219 (0.100)	200.0 (0.0)

Table 4.5: Average (standard deviation) of 25 evolutionary runs with Reber grammar / create a word

Network	Average best individual	Average fitness	Average number of generations
FFN	0.618 (0.009)	0.580 (0.004)	200.0 (0.0)
FFN (tanh)	0.620 (0.007)	0.582 (0.002)	200.0 (0.0)
CTRNN	0.710 (0.047)	0.666 (0.045)	200.0 (0.0)
CTRNN (size changing)	0.674 (0.026)	0.640 (0.024)	200.0 (0.0)
CTRNN (tanh)	0.715 (0.039)	0.675 (0.037)	200.0 (0.0)
CTRNN (tanh, size changing)	0.682 (0.029)	0.649 (0.029)	200.0 (0.0)
GasNet	0.674 (0.017)	0.640 (0.014)	200.0 (0.0)
MSNN (a)	0.627 (0.033)	0.594 (0.032)	200.0 (0.0)
MSNN (b)	0.637 (0.042)	0.604 (0.040)	200.0 (0.0)
MSNN (c)	0.638 (0.037)	0.604 (0.037)	200.0 (0.0)
MSNN (d)	0.628 (0.024)	0.594 (0.022)	200.0 (0.0)
MSNN (full)	0.627 (0.035)	0.595 (0.031)	200.0 (0.0)
MSNN (none)	0.633 (0.039)	0.598 (0.037)	200.0 (0.0)

Table 4.6: Average (standard deviation) of 25 evolutionary runs with embedded Reber grammar / detect a word

Network	Average best individual	Average fitness	Average number of generations
FFN	0.000 (0.000)	0.000 (0.000)	200.0 (0.0)
FFN (tanh)	0.000 (0.000)	0.000 (0.000)	200.0 (0.0)
CTRNN	0.004 (0.012)	0.000 (0.000)	200.0 (0.0)
CTRNN (size changing)	0.003 (0.016)	0.000 (0.000)	200.0 (0.0)
CTRNN (tanh)	0.082 (0.061)	0.006 (0.022)	200.0 (0.0)
CTRNN (tanh, size changing)	0.108 (0.080)	0.036 (0.051)	200.0 (0.0)
GasNet	0.269 (0.071)	0.170 (0.053)	200.0 (0.0)
MSNN (a)	0.064 (0.075)	0.025 (0.045)	200.0 (0.0)
MSNN (b)	0.044 (0.056)	0.008 (0.019)	200.0 (0.0)
MSNN (c)	0.054 (0.061)	0.014 (0.028)	200.0 (0.0)
MSNN (d)	0.042 (0.060)	0.009 (0.024)	200.0 (0.0)
MSNN (full)	0.042 (0.055)	0.009 (0.026)	200.0 (0.0)
MSNN (none)	0.034 (0.057)	0.009 (0.025)	200.0 (0.0)

Table 4.7: Average (standard deviation) of 25 evolutionary runs with embedded Reber grammar / create a word

Chapter 5

Analysis

In the following chapter the evolved modulated spiking neural network will be analysed to get a better understanding of the function, the advantages and especially the problems of the modulated spiking neural network architecture.

5.1 T-Maze

The evolved modulated spiking neural networks (for both the normal and the huge t-maze) show a variety of structures with four or more neurons and different spiking behaviours. One example of an evolved modulated spiking neural network (b modulated) can be seen in 5.3. The structure of the network can be seen in figure 5.1.

In this network neuron 2 does not spike at all because the robot does not need to go backwards. Since it does not have any connections to other neurons and it

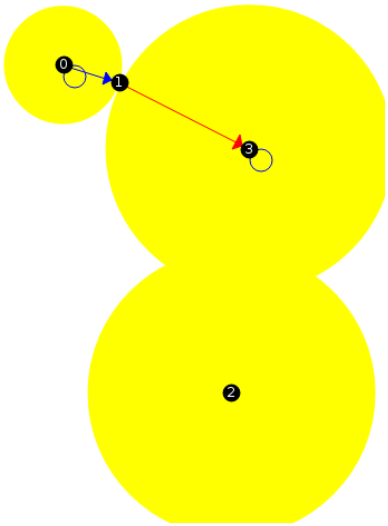
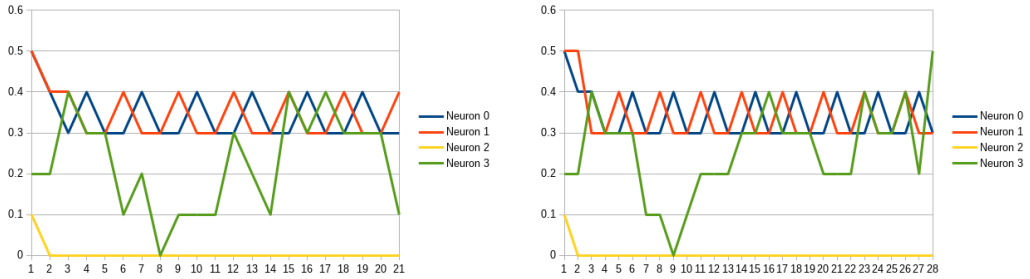


Figure 5.1: Example structure of a modulated spiking neural network (b modulated) solving the t-maze.



(a) MSNN reaching goal 1

(b) MSNN reaching goal 2

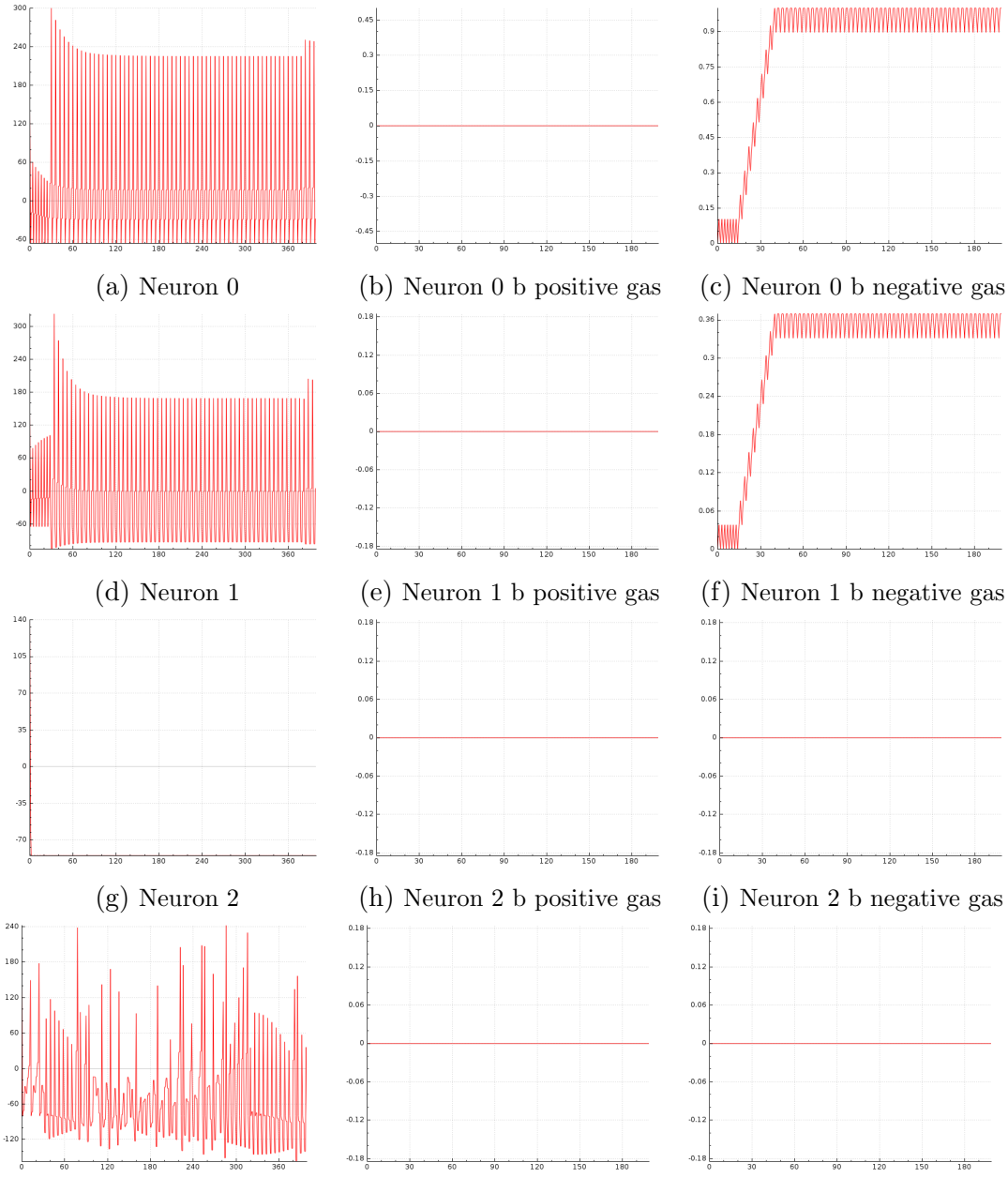
Figure 5.2: Example spiking frequency of a modulated spiking neural network (*b* modulated) solving the t-maze

does not emit any gas, it has no further influence.

Neuron 0 (forward) and neuron 1 (goal 1) always spike at the same frequency while neuron 3 is responsible for choosing the correct goal. If neuron 3 does not receive any input the spiking rate will decrease at the end resulting in the robot taking goal 1 (one example see figure 5.2a), however if neuron 3 receives correct input it will increase the spiking rate resulting in the robot taking goal 2 (one example see figure 5.2b). This way the whole decision is done only by neuron 3.

Since each neuron can only have one input the information of the other input values needs to reach neuron 3 through the other neurons. This is done by the connections $neuron\ 0 \rightarrow neuron\ 1$ and $neuron\ 1 \rightarrow neuron\ 3$ (see figure 5.1). Some examples on how this information is propagated can be seen in figure 5.4, where each line represents a different run. However because this requires the cooperation of multiple neurons it seems to be hard to evolve networks that use all five input values (no network analysed was able to use all input values). The example network was able to use two input values (neuron 1 and 3 used the input of the value 1, neuron 0 the input of value 4). However this seems to be enough to be correct in most cases, because to know the two values are different it is enough to know that you only get one input.

Since the gas is at a constant level, it is not used at all at the network. This is also true for almost every other evolved modulated spiking neural network solving the t-maze, where there is either no gas at all, highly fluctuating gas or gas at a constant level.



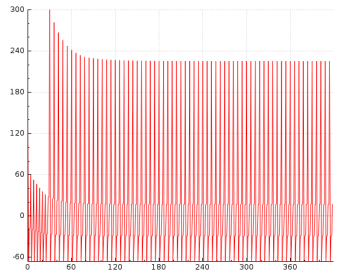
(a) Neuron 0 (b) Neuron 0 b positive gas (c) Neuron 0 b negative gas

(d) Neuron 1 (e) Neuron 1 b positive gas (f) Neuron 1 b negative gas

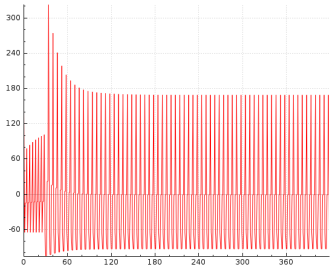
(g) Neuron 2 (h) Neuron 2 b positive gas (i) Neuron 2 b negative gas

(j) Neuron 3 (k) Neuron 3 b positive gas (l) Neuron 3 b negative gas

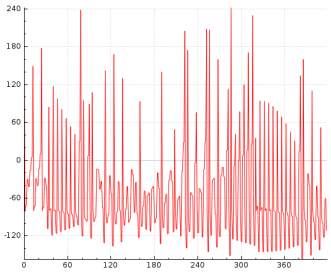
Figure 5.3: Example of a modulated spiking neural network (b modulated) solving the t-maze.



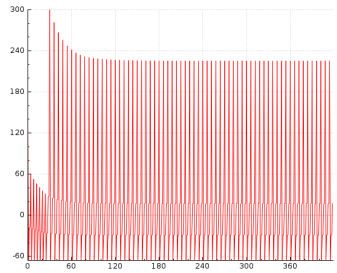
(a) Neuron 0: No input



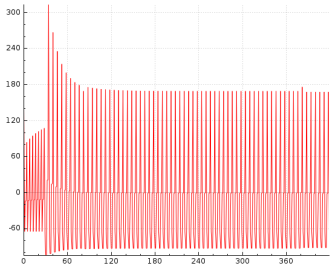
(b) Neuron 1: No input



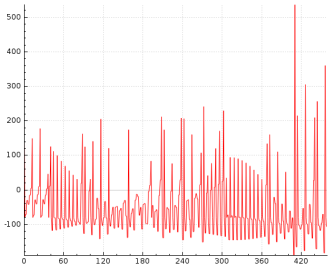
(c) Neuron 3: No input



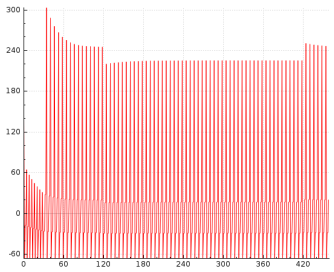
(d) Neuron 0: No input



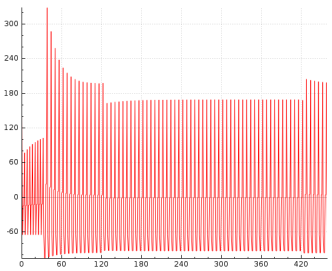
(e) Neuron 1: Input leading to small changes in spiking heights (around 80 and 380)



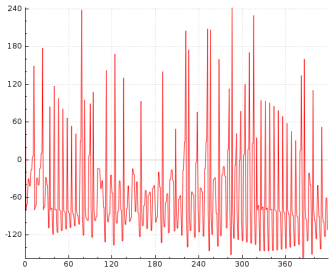
(f) Neuron 3: Input and connection from neuron 1 leading to high spikes at the end and around 110



(g) Neuron 0: Input leading to lower values around 120 and higher spikes around 420



(h) Neuron 1: No input, changes to spiking pattern due to connection from neuron 0



(i) Neuron 3: No input, changes to spiking pattern due to connection from neuron 1

Figure 5.4: Effect of input on example modulated spiking neural network (b modulated) solving the t-maze.

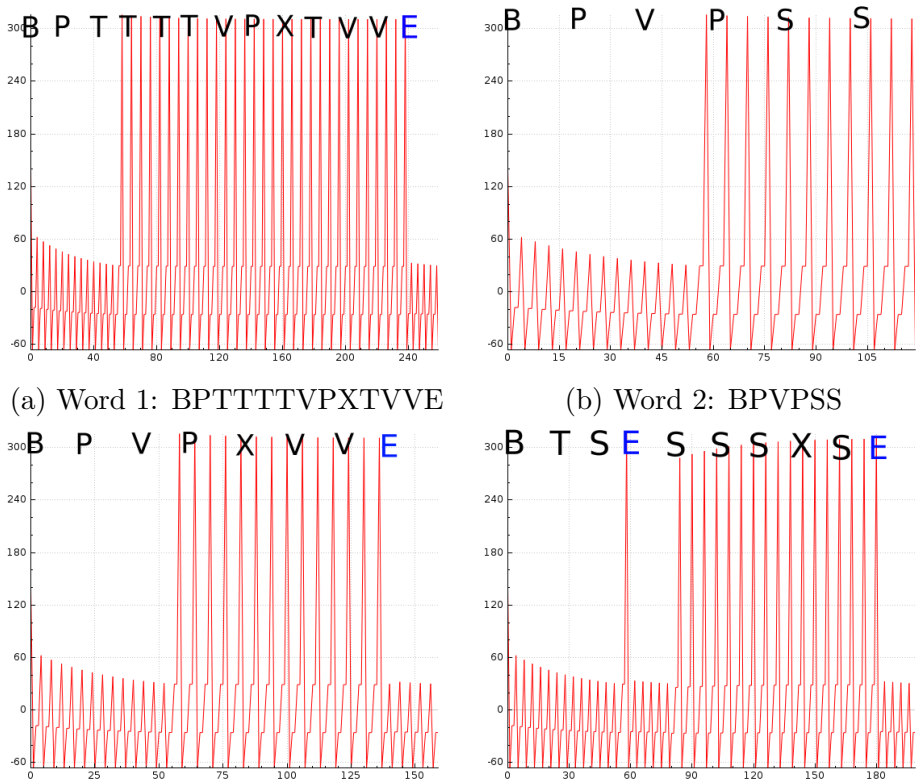
5.2 Reber grammar

5.2.1 Reber detect a word

If you look at the evolved size changing networks (including all modulated spiking neural networks), you can see that neither of them has found a way of detecting correct grammar (for both embedded and non-embedded Reber grammar). One typical example of modulated spiking neural networks can be found in figure 5.5. This network consists of one single neuron which listens for the character **E**. The network only accepts words with the letter **E** at the end, which is slightly better than random guessing.

If you look at the words processed by the network, you can see a constant spiking rate at the beginning, high enough that a word gets accepted. Around time step 60 the height of the spikes suddenly increases - however this has nothing to do with the input. The internal charge of the neuron alternates between negative values and a value slightly over 30 in steps 0 to 60 (approximately $-66 \to 30 \to -66$), which leads to a constant spiking rate of one spike every two steps. This way a word would be accepted. However at about step 60 the spiking changed - instead of reaching a value higher than 30 it reaches a value slightly below 30. The reason for this is that the u value gets increased after each spike (see formula (2.17)), resulting in a smaller charge per step (see formula 2.13). This way one spiking goes approximately $-66 \to 29 \to 300 \to -66$, resulting in a lower spiking rate (one spike every three steps). If no letter **E** is in the word the spiking rate stays at the lower level resulting in the network not accepting the word (e.g. figure 5.5b). However if the letter **E** is put into the network, the additional input is enough to add up to 30 in the second step leading to the original spiking rate. If this is at the end of the word the word is accepted because of the higher spiking rate (e.g. figure 5.5a, 5.5c, 5.5d). Otherwise the spiking rate returns to the lower version after the letter (e.g. figure 5.5d).

In all networks analysed, the gas had no impact at all.



(a) Word 1: BPTTTTVPXTVVE

(b) Word 2: BPVPSS

(c) Word 3: BPVPXVVE

(d) Word 4: BTSESSSXSE

Figure 5.5: Example of neuron activity of MSNN for the "Reber detect a word" simulation including input characters. Characters the network reacts to are coloured blue.

Beginning of word	Completion by network	Shortest legal ending	Correct Reber grammar word
BPTTTVPXT	E	VVE	no
BPTTVP	VE	SE	no
BTX		SE	no
BPTV	V	VE	no
BTXXTTTTVPXV	EEEEE	VE	no
BTSS		XSE	no
BPTTTV	EVVE	VE	no
BPTV	V	VE	no
BPV	VE	VE	yes
BPVP	E	SE	no

Table 5.1: Examples for Reber grammar word completion by a modulated spiking neural network (a modulated)

Beginning of word	Completion by network	Shortest legal ending	Correct Reber grammar word
BTBPV	EETE	VETE	no
BPBPTTV	TE	VEPE	no
BTBPTV	ETE	VETE	no
BTBPV	EETE	VETE	no
BTBPV	EETE	VETE	no
BTBTXXTV	E	VETE	no
BPBPTTTVPXT	T	VVEPE	no
BTBPVP	ETE	SETE	no
BTBTSSSSS	TE	XSETE	no
BPBTXXTTVP		SEPE	no

Table 5.2: Examples for embedded Reber grammar word completion by a modulated spiking neural network (a modulated)

5.2.2 Reber create a word

The modulated spiking neural network tried to learn specific patterns to finish words, however because the grammar is really hard they only developed patterns that work for certain words. Therefore the network tries combinations of the last characters (*VE* or *SE* for Reber grammar, *TE* or *PE* for embedded Reber) to finish the words. Some of the found examples can be seen in tables 5.1 and 5.2.

One interesting aspect that could be observed are feedback loops in which the internal charge of the neuron (normally up to about 400) could get up to really high values. This occurs when a word is longer than the network expected. Some examples of these feedback loops can be seen in figure 5.6. They normally happen when the simulation is running for a longer time than expected by the network (for example because it should complete a long Reber grammar word). These feedback loops could hinder these networks to be used in long running tasks or tasks with highly variable running times.

The reason why these feedback loops occur is a combination of the increase of the u parameter after a spike (see formula (2.17)) and the selection of the parameter ranges, especially the a parameter (see formula (2.18)). When such a feedback loop occurs the function which should reset the u parameter (formula (2.14) and (2.16))

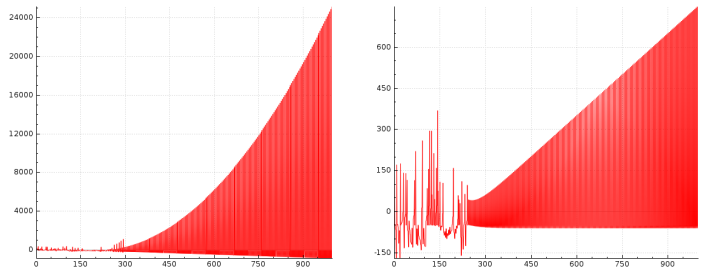
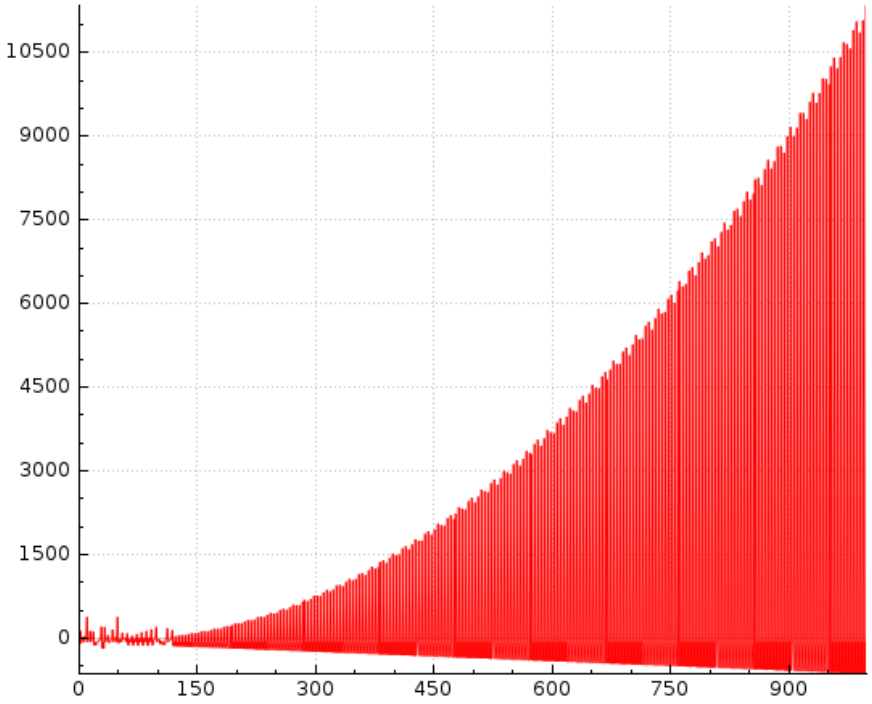


Figure 5.6: Example of feedback loops observed in "Reber create word" task

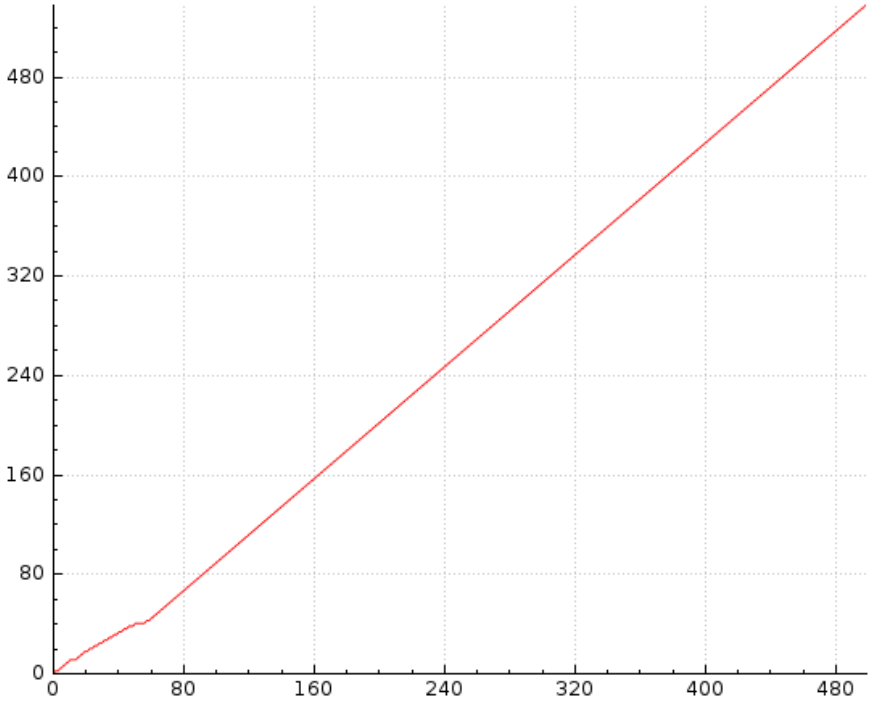
does not work correctly because the a parameter is either zero or very small. This leads to the u parameter growing bigger the more spikes have occurred (see figure 5.7). At one point the value of u has grown so big it is the most prominent factor in the formula for the internal value of a neuron (see formula (2.13) and (2.15)). At this point the spiking cycle turns into the following three steps:

1. Reset and increase of u
2. Highly negative v value because of $-u$ in formula (2.13)
3. Huge v due to v^2 in formula (2.13)
4. Reset and increase of u

This problems comes from the selection of the a parameter by Stefan Bruhns [Bru15]. It can easily be fixed by selecting higher parameters for a.

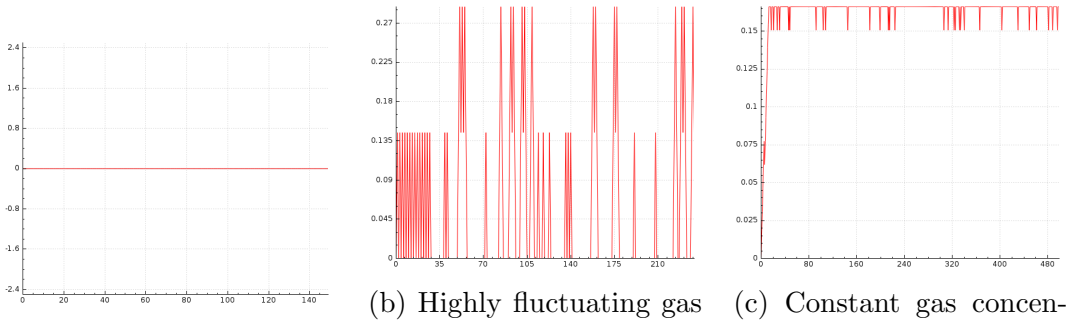


(a) Internal value of neuron (v parameter)



(b) u parameter

Figure 5.7: v and u of one neuron at a feedback loop



(a) No gas emitted

(b) Highly fluctuating gas concentration.

(c) Constant gas concentration

Figure 5.8: Example of gas usage in modulated spiking neural networks

Once again the gas does not seem to have a huge impact on the final networks, with often no gas at all (figure 5.8a), highly fluctuating gas (figure 5.8b) or a constant gas concentration (figure 5.8c). This indicates that the evolution is not able to use the gas mechanism to an advantage. The problem with the observed gas patterns are as follows:

- If *no gas* is emitted at all, then the special feature - the modulation of the neuron behaviour - is never used.
- If the gas is *highly fluctuating* it means the gas has no or only a small impact, especially because it is often emitted and decayed in the same simulation step and thus it can't have a long-time effect.
- If the gas is *constant*, this means we can simply change the value of the modulated parameter of the neuron to get the same effect, thus it is almost equal to having no gas at all.

Chapter 6

Conclusion

The modulated spiking neural network architecture is in general a powerful architecture capable of solving tasks very efficiently (like the t-maze) by using only a few neurons for a lot of computation. We have also looked into the function of the network for different tasks. However, there are some problems with the network architecture that may hold back its potential.

First the way gas is released in modulated spiking neural networks was borrowed from GasNets. The release of gas had a big influence in GasNets [Hus98] and therefore one assumption was that the release of gas would benefit the network - however, because the modulated spiking neural networks work a lot different then traditional networks (like GasNets or CTRNN) due to the spiking neurons, the choice of the gas release function may not be the best one. This problem can probably be solved by finding better conditions for the gas release which are more stable over time. Some possible gas functions may be:

- Instead of depending on values at one internal step (like charge at one internal step or gas concentration at one internal step) it could be useful to use some kind of "moving average" over values (like number of spikes in the last ten time steps or average gas concentration over the last 10 time steps).

- Another method would be to release gas on spikes and only have a small decay when no spike occurs. The decay must be small enough that the gas is not completely decayed if the neuron has a small period of time without spiking.

In addition to that I identified feedback loops as a potential source of problems which can occur when the network is running over a longer period of time, especially if the network does not expect such a long running time. This can be problematic in real world use cases. The problem exists because of the selection of modulation parameter by Stefan Bruhns [Bru15], for which he unfortunately does not give an explanation of his choice. It would be useful to replace them in a way that create biologically meaningful spiking pattern.

After this thesis has analysed the current modulated spiking neural network architecture, the next step would be the improvement of the architecture, especially

the replacement of the gas function and the determination of better modulation parameter. However such work would exceed the scope of this bachelor thesis.

The source code used in this bachelor thesis can be found at *BitBucket* or *GitHub*. See appendix C for more information.

Appendix A

Different Gas Concentration Functions

Husbands et al. used in their original paper the gas concentration function (2.10) [HSJO98, Hus98]. In other works [SHPO02, Smi02, MP06] a different gas concentration function is used:

$$C(d,t) = \begin{cases} C_0 \cdot e^{-(\frac{d}{r})^2} \cdot T_t(t) & d < r \\ 0 & else \end{cases} \tag{A.1}$$

Smith wrote in an annotation in his PhD [Smi02] that the different exponential does not make any significant difference. To verify this I have run some experiments of my own. I have used the t-maze simulation (section 3.2.1) with the genetic algorithm (section 3.1). The results are in table A.1.

The difference between the different gas concentration functions does not show any significant difference for the modulated spiking neural networks (t-test: $P < 5\%$), however there seems to be a significant difference for the GasNets (t-test: $P \leq 5\%$). It would be good to have a more detailed analysis of the impact of the gas concentration function on GasNets, however this lies beyond the scope of this bachelor thesis.

Network	Average best individual	Average fitness	Average number of generations
GasNet (2.10)	0.826 (0.033)	0.782 (0.032)	200.0 (0.0)
GasNet (A.1)	0.857 (0.045)	0.815 (0.046)	198.2 (12.4)
MSNN (full, 2.10)	0.889 (0.056)	0.846 (0.064)	200.0 (0.0)
MSNN (full, A.1)	0.884 (0.051)	0.840 (0.061)	199.0 (6.8)

Table A.1: Average (standard deviation) of 50 evolutionary runs with standard t-maze simulation and different gas concentration functions

Appendix B

Number of generations

All runs presented in the result chapter (see chapter 4) were done with a maximum of 200 generations. The raises the question whether there are any more interesting effects after these 200 generations. To test this I have done some more runs with a maximum of 2000 generations. Therefore I have chosen the best performing modulated spiking neural network (in regards to average best individual) which does not reach an average number of generations other than 200. The results can be seen in table B.1.

In the case of the Reber grammar (both detect a word and create a word) the results are only slightly higher due to the longer optimisation time. However no spikes in fitness or similar events could be observed which lead to the conclusion that a longer running time has no huge impact.

In the case of the t-maze simulation two out of 25 runs reached a network with a success rate higher than 0.99 before reaching 2000 generation. However because of the relative similar fitness values and the high number of networks not reaching 2000 generations we can assume that overall the higher number of generation does not have a huge impact on the results.

Network and simulation	Average best individual	Average fitness	Average number of generations
MSNN (c) t-maze 200	0.909 (0.041)	0.865 (0.054)	200.0 (0.0)
MSNN (c) t-maze 2000	0.944 (0.038)	0.918 (0.043)	1919.0 (351.7)
MSNN (b) Reber create 200	0.343 (0.100)	0.202 (0.097)	200.0 (0.0)
MSNN (b) Reber create 2000	0.472 (0.095)	0.383 (0.093)	2000.0 (0.0)
MSNN (c) Reber detect 200	0.642 (0.034)	0.613 (0.031)	200.0 (0.0)
MSNN (c) Reber detect 2000	0.686 (0.036)	0.663 (0.034)	2000.0 (0.0)

Table B.1: Comparison of 25 evolutionary runs on different simulations with different numbers of generations

Appendix C

Source code listing

The source code used in this bachelor thesis is released under the *GNU Lesser General Public License version 3 or later* (**LGPL3+**) and *GNU Lesser General Public License version 3 or later* (**GPL3+**) on *BitBucket* and *GitHub*. A detailed list can be found at table C.1.

Program	License	BitBucket	Github
qnn library	LGPL3+	https://bitbucket.org/Top-Ranger/qnn	https://github.com/Top-Ranger/qnn
qnn-neuron-visualiser	GPL3+	https://bitbucket.org/Top-Ranger/qnn-neuron-visualiser	https://github.com/Top-Ranger/qnn-neuron-visualiser
qnn-run-analyser	GPL3+	https://bitbucket.org/Top-Ranger/qnn-run-analyser	https://github.com/Top-Ranger/qnn-run-analyser
qnn-single-run	GPL3+	https://bitbucket.org/Top-Ranger/qnn-single-run	https://github.com/Top-Ranger/qnn-single-run
qnn-structure-creator	GPL3+	https://bitbucket.org/Top-Ranger/qnn-structure-creator	https://github.com/Top-Ranger/qnn-structure-creator
qnn-ui	GPL3+	https://bitbucket.org/Top-Ranger/qnn-ui	https://github.com/Top-Ranger/qnn-ui
qnn-visualiser	GPL3+	https://bitbucket.org/Top-Ranger/qnn-visualiser	https://github.com/Top-Ranger/qnn-visualiser

Table C.1: Source code listing

Bibliography

[Bax95] W.G Baxt. Application of artificial neural networks to clinical medicine . *The Lancet*, 346(8983):1135–1138, 1995.

[BRC+07] Romain Brette, Michelle Rudolph, Ted Carnevale, Michael Hines, David Beeman, JamesM. Bower, Markus Diesmann, Abigail Morrison, PhilipH. Goodman, Jr. Harris, FrederickC., Milind Zirpe, Thomas Natschläger, Dejan Pecevski, Bard Ermentrout, Mikael Djurfeldt, Anders Lansner, Olivier Rochel, Thierry Vieville, Eilif Muller, AndrewP. Davison, Sami El Boustani, and Alain Destexhe. Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, 23(3):349–398, 2007.

[Bru15] Stefan Bruhns. Modulated Spiking-Neurons. Bachelor thesis, University of Hamburg, 2015.

[Cox05] Earl Cox. *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration*. Morgan Kaufmann Publishers, 2005.

[Fah91] Scott E. Fahlman. The Recurrent Cascade-Correlation architecture. Technical report, Carnegie Mellon University, 1991.

[HSJO98] Phil Husbands, Tom Smith, Nick Jakobi, and Michael O'Shea. Better Living Through Chemistry: Evolving GasNets for Robot Control. *Connection Science*, 10(3-4):185–210, 1998.

[Hus98] Phil Husbands. Evolving Robot Behaviours with Diffusing Gas Networks. *Lecture Notes in Computer Science*, 1468:71–86, 1998.

[Izh03] Eugene M. Izhikevich. Simple Model of Spiking Neurons. In *IEEE Transactions on neural networks*, number 6 in volume 14, pages 1569–1572, November 2003.

[Izh04] Eugene M. Izhikevich. Which Model to Use for Cortical Spiking Neurons? In *IEEE Transactions on neural networks*, number 5 in volume 15, pages 1063–1070, September 2004.

[Kan11] Mehmed Kantardzic. *Data Mining: Concepts, Models, Methods and Algorithms*. John Wiley & Sons, Inc., 2nd edition, 2011.

[KvdS96] Ben Kröse and Patrick van der Smagt. *An introduction to Neural Networks*. University of Amsterdam, eighth edition, November 1996.

[Maa97] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[MP06] Sven Magg and Andrew Philippides. GasNets and CTRNNs – A Comparison in Terms of Evolvability. In Stefano Nolfi, Gianluca Baldassarre, Raffaele Calabretta, JohnC.T. Hallam, Davide Marocco, Jean-Arcady Meyer, Orazio Miglino, and Domenico Parisi, editors, *From Animals to Animats 9*, volume 4095 of *Lecture Notes in Computer Science*, pages 461–472. Springer Berlin Heidelberg, 2006.

[Ran95] Beer Randall D. *On the dynamics of small continuous-time recurrent neural networks*, volume 03 of *Adaptive Behavior*, pages 469–509. 1995.

[Reb67] Arthur S. Reber. Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behavior*, 6(6):855–863, December 1967.

[SHPO02] Tom Smith, Phil Husbands, Andy Philippides, and Michael O'Shea. Temporally Adaptive Networks: Analysis of GasNet Robot Control Networks. In Russell Standish, Mark A. Bedau, and Hussein A. Abbass, editors, *Artificial Life VIII*, Proceedings of the Eighth International Conference on Artificial Life, pages 274–282, November 2002.

[Smi02] Tom Smith. *The evolvability of artificial neural networks for robot control*. PhD thesis, University of Sussex, November 2002.

[SWE92] J.D. Schaffer, D. Whitley, and L.J. Eshelman. Combinations of genetic algorithms and neural networks: a survey of the state of the art. In *Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92. International Workshop on*, pages 1–37, Jun 1992.

[WRL94] Bernard Widrow, David E. Rumelhart, and Michael A. Lehr. Neural Networks: Applications in Industry, Business and Science. *Commun. ACM*, 37(3):93–105, March 1994.

Erklärung der Urheberschaft

Ich versichere an Eides statt, dass ich die Bachelorarbeit im Studiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum Unterschrift

Erklärung zur Veröffentlichung

Ich erkläre mein Einverständnis mit der Einstellung dieser Bachelorarbeit in den Bestand der Bibliothek.

Ort, Datum Unterschrift